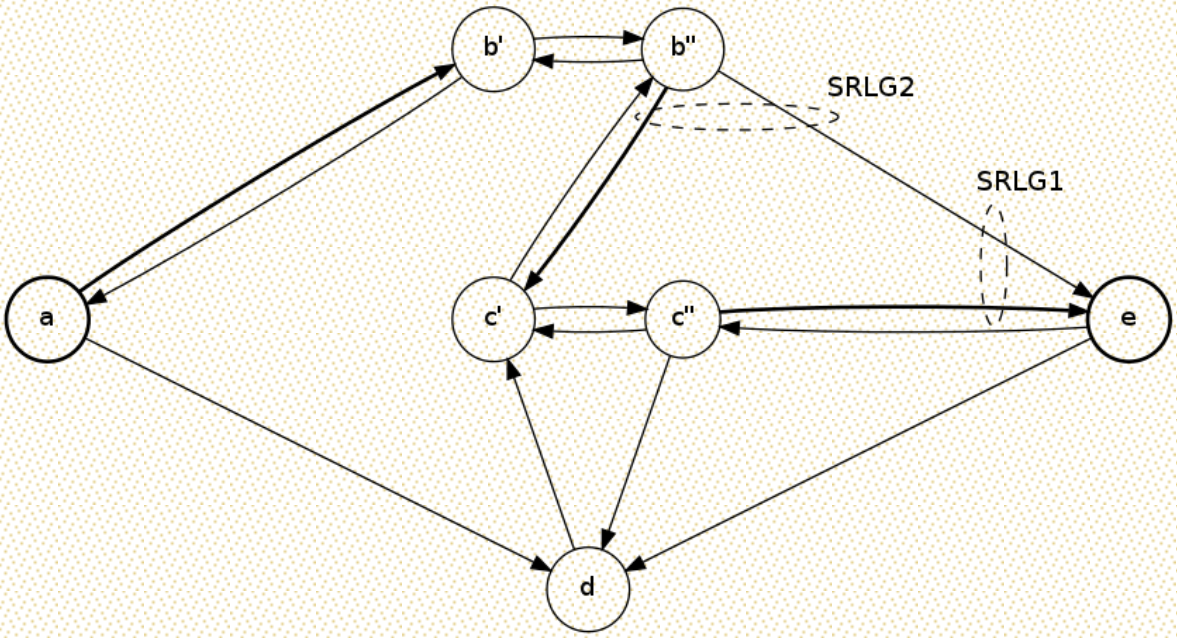

Sérgio Alexandre Figueiredo Mendes

Cálculo de um par de caminhos maximamente disjuntos ou de um par de caminhos disjuntos nas avarias, de custo aditivo mínimo

28 de Janeiro de 2013

· U C ·

Universidade de Coimbra

· U C · FCTUC FACULDADE DE CIÊNCIAS E TECNOLOGIA UNIVERSIDADE DE COIMBRA

FACULDADE DE CIÊNCIAS E TECNOLOGIA
UNIVERSIDADE DE COIMBRA

MESTRADO INTEGRADO EM ENGENHARIA ELECTROTÉCNICA E DE COMPUTADORES

Cálculo de um par de caminhos maximamente disjuntos ou de um par de caminhos disjuntos nas avarias, de custo aditivo mínimo

---

Sérgio Alexandre Figueiredo Mendes

Membros do Júri:
Presidente: Professor Doutor José Manuel Fernandes Craveirinha;
Orientadora: Professora Doutora Teresa Martinez dos Santos Gomes;
Coorientadora: Professora Doutora Rita Cristina Girão Coelho da Silva;
Vogal: Professor Doutor Luís Alberto da Silva Cruz

28 de Janeiro de 2013

Agradecimentos

À Professora Doutora Teresa Martinez Gomes e à Professora Doutora Rita Girão Silva, pela orientação e dedicação.

À Professora Doutora Luísa Jorge e ao Professor Doutor Paulo Melo, pela ajuda, principalmente nos aspectos mais complexos de programação de *software*.

À PT Inovação, em particular ao Engenheiro Vitor Mirones e ao Engenheiro André Brízido, pela disponibilidade e colaboração.

Ao INESC Coimbra, pela atribuição de uma bolsa.

À minha família.

Resumo

Atualmente, com o crescente volume de tráfego em redes de telecomunicações, é de extrema importância a proteção das ligações ponto a ponto estabelecidas ao longo da rede, com o objetivo de evitar interrupções de serviço. Um SRLG (*Shared Risk Link Group*) é um conjunto de elementos da rede que têm um risco comum de falha. Os protocolos de encaminhamento podem distribuir informação acerca dos SRLG que podem afetar cada arco da rede, pelo que se torna importante o desenvolvimento de algoritmos eficientes para a determinação de caminhos disjuntos ou maximamente disjuntos nos SRLG.

Um par de caminhos disjuntos nas avarias é um par de caminhos totalmente disjuntos ou que podem ter em comum elementos resilientes, ou seja que estão protegidos numa camada inferior.

No presente trabalho, desenvolvido no âmbito de um contrato de I&D com a PT Inovação, foram estudados e implementados vários algoritmos: em primeiro lugar um algoritmo de cálculo de um par de caminhos maximamente disjuntos nos nós, de custo aditivo mínimo, que garante que a solução obtida é ótima; em segundo lugar três algoritmos de cálculo de um par de caminhos maximamente disjuntos nos nós e nos SRLG. Cada um desses três algoritmos, propostos no âmbito deste trabalho, são extensões/adaptações de heurísticas para a determinação de pares de caminhos disjuntos nos SRLG; finalmente foi implementada uma heurística, que procura obter um par de caminhos totalmente disjuntos nos nós, exceto em nós extremos de arcos resilientes partilhados por esse par de caminhos.

Os algoritmos foram desenvolvidos tendo em vista a sua utilização em PCE (*Path Computation Element*) integrados em equipamentos de redes GMPLS (*Generalized Multiprotocol Label Switching*). Dado que os PCE integrados têm tipicamente recursos computacionais (capacidade de processamento e quantidade de memória) limitados, procurou-se otimizar os algoritmos implementados.

Foram realizados testes de desempenho das rotinas implementadas, tendo-se verificado que o algoritmo de cálculo de um par de caminhos maximamente disjuntos nos nós, de custo aditivo mínimo, é perfeitamente adequado ao PCE utilizado nos testes. As implementações dos algoritmos de cálculo de um par de caminhos maximamente disjuntos nos nós e nos SRLG, mostraram poder ser utilizadas num PCE no plano de controlo desde que o número de iterações permitido fosse limitado. A última heurística desenvolvida poderá ser utilizada num PCE apenas no plano de gestão uma vez que os tempos de execução não são compatíveis com a sua utilização no plano de controlo, para a rede fornecida pela PT Inovação.

Palavras-Chave: proteção, caminhos maximamente disjuntos, GMPLS, PCE, SRLG

Abstract

Nowadays telecommunication networks face an increasing demand of traffic volume and an increasing need to provide an adequate quality of the service experienced by the users. Therefore the protection of point-to-point connections throughout the network becomes of the utmost importance, in order to avoid service interruptions. A SRLG (*Shared Risk Link Group*) is a set of network elements with common risk of failure. The routing protocols can consider the information on the SRLG affecting each network link. Therefore, the development of efficient algorithms for the calculation of SRLG-disjoint (or at least maximally disjoint) paths becomes a critical issue in this context.

A failure-disjoint path pair is a path pair which is either totally disjoint or only has in common resilient elements (i.e. protected in a lower layer).

In this work, which was developed in the context of a R&D contract with PT Inovação, several algorithms were studied and implemented: firstly, an algorithm for the calculation of a maximally node-disjoint path pair of *min-sum* cost, which guarantees finding an optimal solution; secondly, three algorithms for the calculation of a maximally node and SRLG-disjoint path pair, which are adaptations/extensions of existing heuristics for the calculation of a totally SRLG-disjoint path pair; lastly, a heuristic to calculate a pair of totally node-disjoint paths, except for extreme nodes of resilient links that are shared by that path pair.

The algorithms were developed having in mind that they will be used in a PCE (*Path Computation Element*) in GMPLS (*Generalized Multiprotocol Label Switching*) networks devices, which are usually very limited in terms of computational resources (processing and memory).

Some performance tests for comparison of the implemented algorithms were made. The algorithm for the calculation of maximally node-disjoint path pairs of *min-sum* cost is suitable for the considered PCE. As for the algorithms for the calculation of maximally node and SRLG-disjoint path pairs, they can be used in a PCE as long as the number of allowed iterations is adequate. The heuristic for the calculation of failure-disjoint path pairs can be used in a PCE but only in a management plane due to its long execution time.

Keywords: protection, maximally disjoint paths, GMPLS, PCE, SRLG

Índice

Lista de Figuras

Lista de Tabelas

Acrónimos

BFS	Breadth-First Search
CoSE	Conflicting SRLG Exclusion
CoSE-MS	Conflicting SRLG Exclusion - Min Sum
CPU	Central Processing Unit
DWDM	Dense Wavelength-Division Multiplexing
GMPLS	Generalized Multiprotocol Label Switching
IMSH	Iterative Modified Suurballe's Heuristic
IP	Internet Protocol
LSP	Label-Switched Path
MBH	Modified Bhandari's Heuristic
MBHE	Modified Bhandari's Heuristic Extended
MdCoSE-MS	Modified Conflicting SRLG Exclusion - Min Sum
MdIMSH	Modified Iterative Modified Suurballe's Heuristic
MdTA	Modified Trap Avoidance
MPLS	Multiprotocol Label Switching
MSH	Modified Suurballe's Heuristic
MSHE	Modified Suurballe's Heuristic Extended
OADM	Optical Add/Drop Multiplexer
OXC	Optical Cross-Connect
PCC	Path Computation Client
PCE	Path Computation Element
QoS	Quality of Service
RAM	Random-Access Memory
RHE	Recovery Head-End
RTE	Recovery Tail-End
SRLG	Shared Risk Link Group
TA	Trap Avoidance
TED	Traffic Engineering Database
WDM	Wavelength-Division Multiplexing

Capítulo 1

Introdução

1.1 Motivação

Atualmente, com o crescente volume de tráfego em redes de telecomunicações, é de extrema importância a proteção das ligações ponto a ponto estabelecidas ao longo da rede, com o objetivo de evitar interrupções de serviço. A proteção das ligações é de especial importância em redes óticas baseadas em tecnologias WDM (*Wavelength-Division Multiplexing*) e DWDM (*Dense Wavelength-Division Multiplexing*), uma vez que uma falha, como por exemplo o corte de uma fibra, pode significar a interrupção de várias ligações no domínio ótico e portanto conduzir a um elevado número de utilizadores sem serviço.

Uma das formas de evitar a quebra prolongada de ligações devido a uma eventual falha num dos elementos da rede, é através de proteção global. Essa proteção pode ser conseguida utilizando dois caminhos totalmente disjuntos para a mesma ligação. Um dos caminhos é utilizado enquanto não ocorrer nenhuma falha (caminho ativo), enquanto o outro (caminho de proteção) pode ser utilizado para restabelecer as ligações afetadas pela falha do caminho ativo. Neste cenário é desejável que o caminho ativo e o caminho de proteção sejam totalmente disjuntos ou, se isso não for possível, que sejam maximamente disjuntos, isto é, que partilhem o menor número de elementos da rede, por forma a evitar que uma falha no caminho ativo represente também uma falha no caminho de proteção.

Neste contexto, um conceito importante é o de SRLG (*Shared Risk Link Group*). Um SRLG é um conjunto de elementos da rede que têm um risco comum de falha. Numa rede real um SRLG pode ser por exemplo um conjunto de cabos que passam na mesma conduta, sendo que em caso de destruição da conduta todos os cabos correm o risco de também o ser. Assim, caso a informação acerca dos SRLG a que cada arco pertence esteja disponível, é desejável que o caminho ativo e o caminho de proteção sejam maximamente

disjuntos nos SRLG, isto é, que partilhem o menor número possível de SRLG.

A determinação de um par de caminhos disjuntos nos SRLG é um problema NP-completo [6]. Por esse motivo vários autores apresentaram heurísticas que procuram resolver o referido problema. As heurísticas propostas em [11] e [17] procuram resolver o problema de determinação de um par de caminhos disjuntos nos SRLG formulado como um problema *min-min*, isto é, formulado como um problema em que é pretendido minimizar o custo do caminho ativo. Em [15] e [3], encontram-se heurísticas para resolver o mesmo problema, mas formulado como *min-sum*, isto é, formulado como um problema em que é pretendido minimizar o custo aditivo do par de caminhos.

A heurística IMSH (*Iterative Modified Suurballe's Heuristic*) proposta em [15], recorre a um algoritmo de enumeração de caminhos mais curtos a partir dos quais procura obter um par de caminhos maximamente disjuntos nos SRLG. Em [11] a heurística CoSE (*Conflicting SRLG Exclusion*) procura obter o caminho mais curto em redes em que foram removidos os arcos pertencentes a um dado conjunto de SRLG. Nesse trabalho os autores propõem também uma variante designada de Modified CoSE para obtenção de um par de caminhos maximamente disjuntos, tal que o caminho ativo tem custo mínimo. Em [5] é proposta uma adaptação do CoSE para a resolução do problema *min-sum* designada de CoSE-MS (*Conflicting SRLG Exclusion - Min Sum*), a qual é computacionalmente mais eficiente que o IMSH, mas apresenta soluções de pior qualidade. A heurística *Trap Avoidance* (TA) descrita em [17], obtém o caminho ativo numa rede na qual vai sendo sucessivamente removido um arco, o mais arriscado, identificado como sendo aquele que impediu a obtenção da solução na iteração corrente.

Pode ainda definir-se o conceito de elementos de rede resilientes, que são elementos de rede protegidos numa camada inferior. Neste contexto um par de caminhos disjuntos nas avarias é um par de caminhos totalmente disjuntos ou que podem ter em comum elementos resilientes.

Em [19] é apresentado um algoritmo que resolve em tempo polinomial o problema da determinação de um par de caminhos disjuntos nas avarias, de custo aditivo mínimo, em que o custo de um arco resiliente comum aos dois caminhos do par só é considerado uma única vez.

1.2 Objetivos

Este trabalho foi desenvolvido no âmbito de um contrato de I&D com a PT Inovação e pretende complementar o trabalho apresentado em [13], onde foram desenvolvidos algoritmos para o cálculo de um par de caminhos disjuntos nos nós, de custo aditivo mínimo

e para o cálculo de um par de caminhos disjuntos nos nós e nos SRLG, de custo aditivo mínimo.

Numa dada rede pode não existir um par de caminhos totalmente disjuntos nos nós entre um determinado nó de origem e um determinado nó de destino. No entanto pode existir um par de caminhos que tenha apenas alguns arcos ou nós (intermédios) em comum.

O primeiro objetivo deste trabalho foi a implementação de um algoritmo para o cálculo de um par de caminhos maximamente disjuntos nos nós, de custo aditivo mínimo. Caso existam pares de caminhos totalmente disjuntos, o algoritmo devolve o que tiver menor custo aditivo.

Os SRLG são utilizados para permitir a uma camada superior da rede obter caminhos disjuntos nas falhas da camada inferior, sem que os detalhes desta sejam comunicados à primeira. Neste contexto, o segundo objetivo deste trabalho foi a implementação de heurísticas para a determinação de um par de caminhos maximamente disjuntos nos nós e nos SRLG, procurando também minimizar o custo aditivo do par.

O terceiro objetivo foi a implementação de um algoritmo de cálculo de um par de caminhos disjuntos nos nós e nas avarias, de custo aditivo mínimo, ou seja que apenas partilhem arcos resilientes (e nesse caso os respetivos nós extremos).

Um objetivo comum a todas as implementações realizadas foi a obtenção de rotinas tão eficientes quanto possível, para tornar viável a sua utilização num PCE (*Path Computation Element*) com recursos computacionais (nomeadamente CPU e memória) limitados. Foi por isso testado o desempenho relativo dos algoritmos, para além da averiguação da qualidade das soluções produzidas, utilizando em particular uma rede de teste fornecida pela PT Inovação.

1.3 Conteúdo

No capítulo 2 é feita a contextualização do trabalho desenvolvido e são apresentados alguns conceitos e definições. No capítulo 3 é apresentado o algoritmo para o cálculo de um par de caminhos maximamente disjuntos nos nós, de custo aditivo mínimo. No capítulo 4 são apresentados três algoritmos para o cálculo de um par de caminhos maximamente disjuntos nos nós e nos SRLG. No capítulo 5 é apresentado um algoritmo para o cálculo de um par de caminhos disjuntos nas avarias, de custo aditivo mínimo. No capítulo 6 são apresentados resultados de desempenho dos algoritmos implementados e é analisada a qualidade relativa das soluções produzidas. Por fim, no capítulo 7 são apresentadas algumas conclusões e trabalho futuro.

Capítulo 2

Contextualização do trabalho

2.1 Camada física/Camada ótica

Uma rede ótica pode ser vista como composta por duas camadas, a camada física e a camada ótica [6]. A camada física é constituída pelos componentes físicos de ligação, por exemplo cabos e condutas, pelos nós de comutação ótica, por exemplo OADM (*Optical Add/Drop Multiplexers*) e OXC (*Optical Cross-Connects*), e pelos equipamentos terminais. A camada ótica consiste nas ligações efetuadas no domínio ótico.

Uma ligação na camada ótica pode atravessar vários componentes da camada física e várias ligações na camada ótica podem passar através dos mesmos componentes da camada física, de tal forma que uma falha, como por exemplo a rotura de um cabo de fibra ótica, pode causar a falha das múltiplas ligações no domínio ótico.

Para proteger uma ligação na camada ótica entre dois nós de uma rede são tipicamente determinados dois caminhos diferentes, na camada física. Um deles é o chamado caminho ativo, que é usado em condições normais, enquanto o outro é o chamado caminho de proteção, que é usado em caso de falha do caminho ativo. Para evitar uma quebra de conexão, devido a uma falha isolada, o caminho de proteção não deve partilhar, se possível, nenhum nó (e portanto nenhum arco) com o caminho ativo.

2.2 GMPLS (*Generalized Multiprotocol Label Switching*)

O MPLS (*Multiprotocol Label Switching*) é uma tecnologia para envio de pacotes que utiliza etiquetas para o seu encaminhamento. À entrada da rede MPLS os pacotes são etiquetados e são de seguida encaminhados por caminhos designados por LSP (*Label-Switched Path*). Em cada nó, a decisão de encaminhamento é feita com base na etiqueta de entrada, sendo o pacote reexpedido com uma nova etiqueta (comutação de etiquetas).

Os principais objetivos do MPLS são a eficiência da utilização de largura de banda, garantir níveis estritos de QoS (*Quality of Service*) para diferentes serviços, assim como garantir a recuperação rápida de situações de falha [16].

O GMPLS (*Generalized Multiprotocol Label Switching*) procura aumentar as funcionalidades do MPLS englobando no seu plano de controlo (sinalização e encaminhamento), para além de equipamentos de comutação de pacotes, equipamentos de comutação no tempo, comutação no comprimento de onda e comutação na fibra. O principal objetivo da utilização de um plano de controlo comum é procurar simplificar o funcionamento e a gestão da rede através do estabelecimento automático de ligações ponto a ponto, garantindo a qualidade de serviço desejada.

2.3 PCE (*Path Computation Element*)

O cálculo de caminhos entre dois nós de uma rede é um processo fundamental em redes GMPLS.

Um PCE [2] é um elemento que faz o cálculo de caminhos numa rede, com base na informação contida numa base de dados designada por TED (*Traffic Engineering Database*) que contém informação tal como a topologia da rede a que está associada.

Um PCE pode ser parte integrante de um nó da rede, por exemplo pode fazer parte de um *router* (o PCE utilizado para realizar os testes de desempenho dos algoritmos desenvolvidos é deste tipo), ou pode ser um servidor dedicado.

Uma arquitetura de rede com PCE pode ser baseada num modelo computacional centralizado, em que todos os cálculos de caminhos são efetuados por um único PCE, isto é, todos os PCC (*Path Computation Client*) enviam os seus pedidos para cálculo de caminhos para um PCE central. Pode ser também baseada num modelo computacional distribuído, em que existem múltiplos PCE e onde o cálculo de caminhos é partilhado pelos PCE, sendo que um dado caminho pode ser calculado por um único PCE ou vários PCE (um PCC pode estar ligado a um PCE em particular ou pode ser livre de escolher de entre vários PCE).

Este trabalho foi desenvolvido no âmbito de um contrato de investigação e desenvolvimento em colaboração com a PT Inovação para posterior incorporação das rotinas desenvolvidas na pilha protocolar a instalar num PCE. Por esse motivo foi preciso ter em especial atenção a utilização de memória e do tempo de execução num PCE (UNICOM-V5) disponibilizado pela PT Inovação. Em particular, foi utilizada a representação da rede definida no âmbito da dissertação [13], também ela elaborada no âmbito de um contrato

de investigação e desenvolvimento em colaboração com a PT Inovação. Nessa representação foi previsto um campo para ativar/desativar arcos. Assim sempre que é necessário realizar uma transformação da rede, em vez de duplicar toda a representação da mesma, procede-se simplesmente à ativação ou desativação dos arcos necessários. Este tipo de manipulações surge ao longo dos algoritmos apresentados, tal como por exemplo no algoritmo 1, em que na linha 8 é feita a transformação do grafo correspondente ao algoritmo auxiliar 16 no anexo A.2.1 e na linha 10 é reposto o estado original da rede.

2.4 Recuperação de falhas

A recuperação de falhas em redes de telecomunicações pode ser pré-planeada ou dinâmica [16]. Em recuperação pré-planeada, também designada por proteção, são calculados caminhos de proteção antes da ocorrência de uma falha, enquanto que na recuperação dinâmica, ou restauro, os caminhos de recuperação são calculados apenas após a ocorrência de alguma falha no caminho ativo. A recuperação pré-planeada tem a vantagem de ter um tempo de recuperação mais baixo, enquanto que a recuperação dinâmica necessita de tempo adicional para calcular o caminho de recuperação.

A recuperação pode ainda ser classificada em dois tipos, quanto à sua extensão, recuperação local ou recuperação global. Seja o nó de origem do desvio de recuperação designado por RHE (*Recovery Head End*) e o nó de destino desse mesmo desvio designado por RTE (*Recovery Tail End*). Na recuperação local apenas os elementos do caminho ativo afetados por uma falha são evitados, ou seja, o RHE e o RTE são escolhidos tão próximos dos elementos que falharam quanto possível. Na recuperação global, idealmente, é evitada a totalidade do caminho ativo entre origem e destino, ou seja, o RHE e o RTE irão coincidir com a origem e o destino do caminho ativo respetivamente.

Existem os seguintes esquemas de proteção [16]:

1+1 (proteção dedicada):

Um caminho de proteção dedicado protege um caminho ativo (ou um segmento do caminho ativo). O tráfego é duplicado no RHE e é enviado pelo caminho ativo e pelo caminho de proteção. No RTE o sinal de melhor qualidade é selecionado, ou alternativamente pode ser selecionado o sinal do caminho ativo e apenas caso este não cumpra determinadas condições será selecionado o sinal do caminho de proteção. Este tipo de proteção é eficiente em termos de tempo de recuperação, mas ineficiente em termos de utilização de largura de banda.

1:1 (proteção dedicada com tráfego adicional):

Um caminho de proteção dedicado protege um caminho ativo (ou um segmento do caminho ativo). Na ausência de falhas o tráfego é transportado apenas pelo caminho ativo, o que permite transportar tráfego adicional através do caminho de proteção. Quando é detetada uma falha no caminho ativo o tráfego adicional é retirado do caminho de proteção, sendo substituído pelo tráfego afetado pela falha do caminho ativo.

1:N (recuperação partilhada com tráfego adicional):

Um caminho de proteção é dedicado à proteção de até N caminhos ativos. Na ausência de falhas o caminho de proteção pode ser utilizado para transportar tráfego adicional.

M:N (com M $\leq$ N):

Um conjunto de M caminhos de proteção protege um conjunto de até N caminhos ativos específicos. Pode ser transportado tráfego adicional através dos M caminhos de proteção, quando disponíveis.

2.5 Conceitos e Notação

No trabalho desenvolvido recorre-se à representação de redes através de grafos. Por esse motivo os termos grafo e rede serão utilizados, a partir daqui, como tendo o mesmo significado.

Os grafos podem ser classificados em dois tipos, grafos dirigidos ou grafos não dirigidos.

Um grafo dirigido $G = (V,A)$ é constituído por um conjunto de nós e por um conjunto de arcos, representando V o conjunto dos nós e A o conjunto dos arcos. Cada arco liga dois nós numa determinada ordem, isto é, cada arco forma um par ordenado de nós representado genericamente por $(i, j), i, j \in V$. O número de nós presentes num grafo dirigido pode então ser representado por $|V|$ e o número de arcos representado por $|A|$. A figura 2.1 é um exemplo de um grafo dirigido, com $V = \{a,b,c,d,e\}$ e $A = \{(a,b),(a,d),(b,a),(b,c),(b,e),(c,d),(c,e),\ (d,c),(d,e),(e,d)\}$, onde os nós são representados por círculos e os arcos são representados por setas. O custo de cada arco está indicado na sua proximidade.

Um grafo não dirigido $G = (V,E)$ é constituído por um conjunto de nós e por um conjunto de arestas, representando V o conjunto dos nós e E o conjunto de arestas. As arestas não têm uma direção associada, isto é, a aresta (i, j) é idêntica à aresta (j,i).

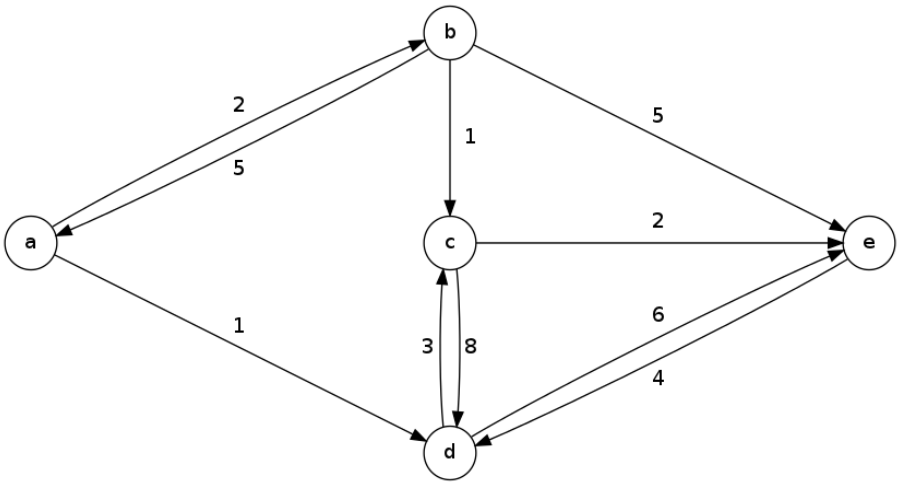

Figura 2.1: Grafo representativo de uma rede.

A figura 2.2 é um exemplo de um grafo não dirigido, com $V = \{a,b,c,d,e\}$ e $E = \{(a,b),(a,d),(b,c),(b,e),(c,d),(c,e),(d,e)\}$.

A figura 2.3 é uma representação do grafo não dirigido da figura 2.2, através de um grafo dirigido.

No trabalho desenvolvido apenas as redes dirigidas têm interesse, por isso, a partir daqui, a utilização da palavra rede terá implicitamente o significado de rede dirigida, salvo indicação em contrário.

Nas definições que se seguem considera-se que $l(i,j)$ é o custo do arco (i,j), $d(i,j)$ é o custo do caminho de i para j, e p e q são dois caminhos de s para t num grafo $G = (V,A)$, com $i,j,s,t \in V$ e $(i,j) \in A$. Note-se que $l(i,j)$ é por omissão inteiro e estritamente positivo.

- **Cabeça de um arco:** é o nó apontado por esse arco. A cabeça do arco definido pelo par ordenado de nós (i,j) é o nó j.
- **Cauda de um arco:** é o nó de onde parte esse arco. A cauda do arco definido por (i,j) é o nó i.
- **Arco simétrico:** o arco simétrico de um arco (i,j) é o arco (j,i).
- **Nós vizinhos:** os nós vizinhos de um nó i podem ser definidos por:
  $\Gamma_i = \Gamma_i^+ \cup \Gamma_i^- :$
  $\Gamma_i^+ = \{u \in V \setminus \{i\} : (i,u) \in A\}$ (nós que são cabeça de um arco cuja cauda é i)
  $\Gamma_i^- = \{w \in V \setminus \{i\} : (w,i) \in A\}$ (nós que são cauda de um arco cuja cabeça é i)
- **Caminho:** o caminho p do nó s para o nó t pode ser visto como uma sequência de arcos que ligam o nó s ao nó t. Como os arcos de um grafo dirigido são definidos

à custa de pares ordenados de nós, um caminho também pode ser visto como uma sequência de nós. Por exemplo, os caminhos entre o nó a e o nó e, do grafo da figura 2.1, podem ser definidos por abe, $abce$, $adce$, ade e $abcde$. O conjunto dos nós de um caminho p será designado por V_p e o conjunto dos arcos desse mesmo caminho por A_p.

- **Segmento:** é uma qualquer sequência de nós consecutivos de um caminho. Dado um caminho representado pela sequência de nós $v_1 v_2 \dots v_n$, um segmento deste caminho é uma qualquer sequência do tipo: $v_k v_{k+1} v_{k+2} \dots v_{k+m}$, com $k+m \leq n$ e $k \geq 1$.

- **Nó predecessor:** o nó predecessor do nó i, $P(i)$, é aquele que, na sequência de nós que define um caminho, se encontra na posição imediatamente anterior a i.

- **Nó intermédio de um caminho:** se s e t forem o nó de origem e o nó de destino, respetivamente, de um caminho p, os nós intermédios do caminho são aqueles pertencentes ao conjunto $V_p \backslash \{s,t\}$.

- **Caminho sem ciclos:** é um caminho em que nenhum nó aparece mais do que uma vez. Neste projeto apenas serão considerados caminhos deste tipo.

- **Custo de um caminho p de s para t:** é igual à soma dos custos de cada um dos arcos que compõem esse caminho. $d(s,t) = d(p) = \sum l(i,j) : (i,j) \in p$, em que s representa o nó de origem do caminho p e t representa o nó de destino do caminho p.

- **Caminho mais curto:** o caminho mais curto do nó i para o nó j é aquele cujo custo é menor. Pode existir mais do que um caminho mais curto.

- **Ciclo:** é cada um dos caminhos possíveis do nó i para si mesmo, em que o único nó repetido é i. Um ciclo diz-se negativo se $d(i,i) < 0$.

- **Caminhos disjuntos:** dois ou mais caminhos, de s para t, dizem-se disjuntos (ou totalmente disjuntos), nos nós ou nos arcos, se não têm em comum nenhum nó intermédio ou arco, respetivamente. Dois caminhos p e q são disjuntos se verificarem $V_p \cap V_q = \{s,t\}$. Note-se que a existência de disjunção nos nós implica a existência de disjunção nos arcos, mas o recíproco não é necessariamente verdade.

- **Caminhos maximamente disjuntos:** dois ou mais caminhos dizem-se maximamente disjuntos, de s para t, nos nós ou nos arcos, se têm em comum o menor número possível de nós ou arcos, respetivamente. Esta definição implica que dois

caminhos disjuntos são também maximamente disjuntos (mas o recíproco não é verdade).

- **Custo aditivo de um par de caminhos maximamente disjuntos p,q, entre s e t:** é a soma do custo de cada um dos caminhos que formam o par, ou seja, $d(p,q) = d(p)+d(q)$.

- **Custo aditivo de um par de caminhos disjuntos nas avarias p,q, entre s e t:** Seja o conjunto dos arcos resilientes na rede dado por $\bar{S}$. Seja $\bar{S}_{(p,q)} = A_p \cap A_q$, onde $A_p \cap A_q \subseteq \bar{S}$. Então, $d(p,q) = d(p)+d(q)-\sum_{k\in\bar{S}_{(p,q)}} l_k$

  Nesta definição, o custo de eventuais arcos resilientes comuns aos dois caminhos só é contabilizado uma vez no cálculo do custo do par.

- **SRLG (Shared Risk Link Group):** é um conjunto de arcos que têm risco de falhar simultaneamente. Seja R o conjunto de todos os SRLG de uma dada rede. Pode definir-se: $R = \{r_\rho : \rho \in \Upsilon\}$, em que r_ρ é o conjunto dos arcos da rede afetados pelo risco ρ e Υ é o conjunto de todos os riscos presentes na rede.

- **Nó de articulação:** é um nó cuja remoção desconecta uma rede não dirigida.

- **Ponte:** é uma aresta cuja remoção desconecta uma rede não dirigida.

- $\mathscr{P}_{st}$**:** é o conjunto de todos os caminhos de s para t.

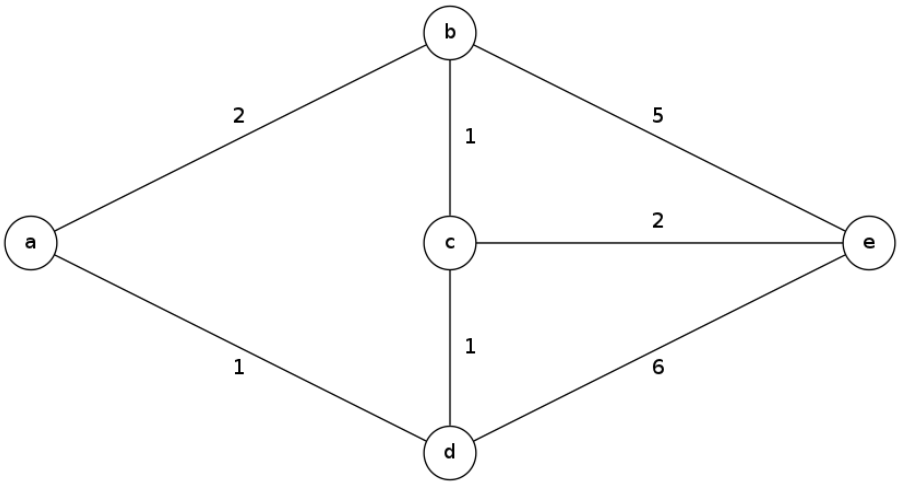

Figura 2.2: Exemplo de um grafo não dirigido.

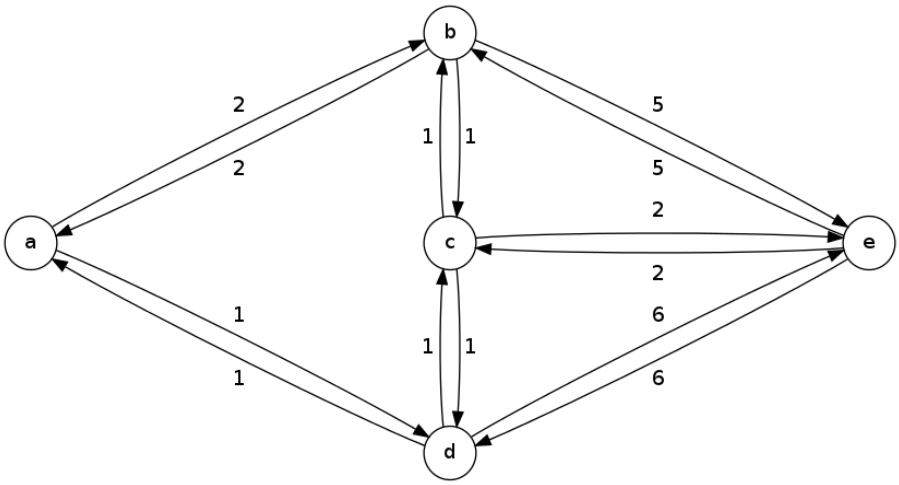

Figura 2.3: Representação do grafo não dirigido da figura 2.2 através de um grafo dirigido.

Capítulo 3

Algoritmo para a determinação de um par de caminhos maximamente disjuntos nos nós, de custo aditivo mínimo

3.1 Introdução

Seja uma rede representada pelo grafo $G = (V,A)$, sendo o custo de utilização de um arco $(i, j) \in A$ dado por $l(i, j)$ e s e t o par de nós entre os quais se pretende obter um par de caminhos disjuntos nos nós de custo aditivo mínimo, seja ele (p^*, q^*). Este problema pode ser formalizado da seguinte forma [13]:

$$(p^*, q^*) = \arg \min_{p,q \in \mathscr{P}_{st}} d(p) + d(q) \tag{3.1}$$

$$\text{s.a}: \quad V_p \cap V_q = \{s,t\} \tag{3.2}$$

$$p \neq q \tag{3.3}$$

O problema anterior é resolvido em tempo polinomial pelos algoritmos de Suurballe ou Bhandari [1].

Neste capítulo pretende-se obter um par de caminhos maximamente disjuntos nos nós e nos arcos, tal que o seu custo seja mínimo. Isto corresponde a resolver um problema

multicritério, para o qual se definem as seguintes funções objetivo:

$$f_1(p,q) = |(A_p \cap A_q)| \tag{3.4}$$
$$f_2(p,q) = |(V_p \cap V_q) \backslash \{s,t\}| \tag{3.5}$$
$$f_3(p,q) = d(p) + d(q) \tag{3.6}$$

A função $f_1(p,q)$ devolve o número de arcos comuns a p e q, a função $f_2(p,q)$ devolve o número de nós intermédios comuns a p e q e a função $f_3(p,q)$ devolve o custo aditivo do par de caminhos p e q.

Pode considerar-se uma abordagem lexicográfica para a ordenação das soluções deste problema, em que cada um dos problemas de otimização é resolvido sequencialmente [8]:

$$\min_{p,q \in \mathscr{P}_{st}} f_i(p,q) \tag{3.7}$$

sujeito a $p \neq q$ e $f_j(p,q) \leq f_j(p_j^*, q_j^*)$, $j = 1,2,\ldots i-1$ $(i > 1)$, $i = 1,2,3$, em que f_i é a i-ésima função objetivo e i representa a posição (ordem de importância) de uma função na sequência de preferência. O par de caminhos (p_j^*, q_j^*) representa o ótimo da j-ésima função obtida na j-ésima iteração. Depois da primeira iteração $(j = 1)$ $f_j(p_j^*, q_j^*)$ não tem necessariamente o mesmo valor que o mínimo independente de $f_j(p,q)$, devido às novas restrições.

Uma solução (p', q') domina lexicograficamente a solução (p,q), no problema (3.7), se $f_k(p', q') < f_k(p,q)$ com $k = \min_j : f_j(p,q) \neq f_j(p', q')$.

Em [1] é descrito um algoritmo para o cálculo de um par de caminhos maximamente disjuntos nos arcos e nos nós, de custo aditivo mínimo. Bhandari prova que esse algoritmo [1, algoritmo 4.2, pág. 104] devolve o par de caminhos com menor número de nós e arcos em comum e cujo custo aditivo é mínimo (solução ótima). Este algoritmo foi estudado e implementado no âmbito deste trabalho e encontra-se descrito na secção seguinte.

3.2 Algoritmo para o cálculo de um par de caminhos maximamente disjuntos

A ideia base do algoritmo de cálculo de um par de caminhos maximamente disjuntos é semelhante à do algoritmo de obtenção do par de caminhos disjuntos nos arcos de custo aditivo mínimo. Para obter um par de caminhos disjuntos nos arcos de custo aditivo mínimo usando o algoritmo de Bhandari [1], são necessários essencialmente quatro passos:

determinar o caminho mais curto do nó origem para o nó de destino; transformar a rede, removendo os arcos desse caminho da rede e substituindo-os pelo seu simétrico fazendo o seu custo tomar o valor simétrico correspondente; nesta rede transformada calcular o caminho mais curto; o par resultante é obtido após a remoção dos arcos simétricos de arcos do primeiro caminho presentes no segundo.

Se for desejado obter um par de caminhos disjuntos nos nós de custo aditivo mínimo, tudo se passa como anteriormente exceto no passo de transformação da rede. Neste caso é necessário começar por dividir os nós intermédios do primeiro caminho obtido, de forma a que estes passem a ser representados por um arco. Para tal, cada nó i deste caminho será representado por dois nós, um nó i' e um nó i'', ligados por um arco (i'', i') de custo 0. Os arcos externos ao caminho que eram incidentes em i passam a ser incidentes em i' e os arcos externos que eram emergentes de i passam a ser emergentes de i''.

A remoção dos arcos, no sentido da origem para o destino, efetuada quando se pretende obter pares de caminhos disjuntos, garante que o segundo caminho não tem nenhum arco (nó) em comum com o primeiro. No presente caso pretende-se obter uma solução mesmo quando a disjunção total não é possível, pelo que estes arcos não podem ser removidos. Por outro lado pretende-se que estes só sejam utilizados se não for possível evitá-los. Assim na transformação da rede estes arcos permanecem, mas com um custo suficientemente elevado que garante que apenas são utilizados se forem nós ou arcos que separam s de t.

A utilização de um nó em comum será penalizada pelo custo y, a atribuir ao arco (i', i''), tal que $y > \sum_{k \in A} l_k$, ou seja, $y = \sum_{k \in A} l_k + \varepsilon$, com $\varepsilon > 0$ (em que k representa um arco $(i, j) \in A$, para simplificar a notação). Este valor garante que y é superior ao custo de qualquer caminho de s para t na rede. Considera-se ainda a constante $x = (|V| - 1)y$, a qual será adicionada ao custo de cada arco do caminho mais curto por forma a evitar a sua utilização. O valor de x é superior ao de y, o que garante preferencialmente proteção aos arcos sempre que possível. Como já foi referido, a utilização destas constantes no algoritmo 1 garante a obtenção de um par de caminhos maximamente disjuntos nos nós, de custo aditivo mínimo. O algoritmo 16 (apresentado no anexo A.2.1) traduz toda esta operação.

O algoritmo [1] procura obter um par de caminhos disjuntos em redes não dirigidas. Se existir um par de caminhos totalmente disjuntos, será obtido o correspondente par disjunto de custo aditivo mínimo. A descrição que segue é uma versão ligeiramente modificada do algoritmo [1, algoritmo 4.2, pág. 104], de forma a funcionar corretamente em redes dirigidas, através de uma versão modificada da transformação da rede no algoritmo 16 (ver anexo A.2.1).

Algoritmo 1: Algoritmo de Bhandari para cálculo de um par de caminhos maximamente disjuntos nos nós, de custo aditivo mínimo

Dados: Grafo $G = (V,A)$, nó de origem s, nó de destino t, custo de cada arco l_k com $k \in A$

Resultado: Par de caminhos maximamente disjuntos nos nós, de custo aditivo mínimo, entre s e t

1 $y \leftarrow L + \varepsilon$ //Em que $L = \sum_{k \in A} l_k$ e $\varepsilon > 0$
2 $x \leftarrow (|V| - 1)y$ //Em que $|V|$ é o número de nós da rede
3 $p' \leftarrow \text{shortestPath}(s,t,G)$ //Determinar o caminho mais curto de s para t na rede, utilizando o algoritmo de Dijkstra ou o algoritmo BFS (ver anexo A.1)
4 **se** $p' = \emptyset$ **então**
5 $\quad (p,q) \leftarrow (\emptyset,\emptyset)$
6 $\quad$ Termina // Sem solução
7 **fim**
8 Transformar a rede (G) de acordo com o algoritmo 16 (ver anexo A.2.1)
9 $q' \leftarrow \text{shortestPath}(s,t,G)$ //Determinar o caminho mais curto de s para t, na rede modificada, usando o algoritmo de Dijkstra modificado ou o algoritmo BFS
10 Repor a rede (G) na sua forma original antes da transformação efetuada na linha 8
11 $(p,q) \leftarrow \text{removeInterlacing}(p',q')$ //Remover os arcos dos dois caminhos mais curtos determinados, que sejam simétricos – algoritmo 17 (ver anexo A.2.2)
12 **se** $p = q$ **então**
13 $\quad (p,q) \leftarrow (\emptyset,\emptyset)$
14 $\quad$ //Sem solução, apenas existe um caminho de s para t
15 **fim**

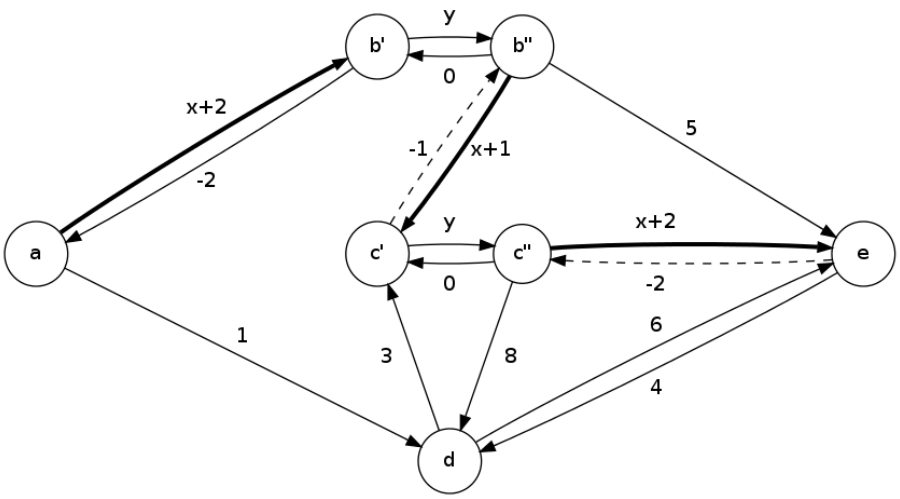

Figura 3.1: Representação da rede da figura 2.1 após a divisão dos nós (*node-splitting*), de acordo com o algoritmo 16 (anexo A.2.1).

A título de exemplo, considere-se a execução do algoritmo na rede representada na figura 2.1 considerando o nó a como nó de origem e o nó e como nó de destino. Primeiro é determinado o caminho mais curto do nó a para o nó e na rede, que é o caminho $abce$ com custo 5. Depois é feita a divisão dos nós do caminho $abce$ de acordo com o algoritmo 16 (ver A.2.1), ficando a rede na forma representada na figura 3.1 (onde os arcos que fazem parte do caminho $abce$ estão representados com traço mais espesso).

Note-se que foram criados os arcos (c',b'') e (e,c'') (representados na figura 3.1 a tracejado), uma vez que esses arcos não existiam (A.2.1). Note-se ainda que quer o arco (b',a) quer o arco (e,c'') nunca farão parte do caminho mais curto (sem ciclos) entre o nó a e o nó e, ou seja, nesta rede modificada poderiam omitir-se o arco incidente no nó origem e o arco emergente do nó destino resultantes da transformação da rede. Contudo na descrição dos algoritmos com este tipo de transformação nunca será referido este facto para simplificar a descrição da transformação, aliás como é feito em geral na literatura. De igual forma nas figuras ilustrativas de transformações de rede neste capítulo e nos seguintes, esses arcos também não serão omitidos. Isto não obriga a que a implementação da transformação de rede inclua a criação desses arcos; em particular na implementação realizada e tendo em conta a estrutura de dados usada na representação da rede (analisada em detalhe em [13]), o arco emergente do nó destino nunca é criado.

O algoritmo de Dijkstra (A.1.1) não pode ser usado para calcular o caminho mais curto na rede modificada (isto é, na rede resultante da execução do algoritmo da divisão dos nós) pois o algoritmo da divisão dos nós dá origem a arcos de custo negativo. Então tem de ser usada uma versão modificada do algoritmo de Dijkstra (A.1.2) ou o algoritmo BFS

(A.1.3). No exemplo, o caminho mais curto na rede modificada (figura 3.1) é *ade* com custo 7. Como o caminho *ade* não usa nenhum arco simétrico de algum arco do caminho mais curto *abce* não é necessária a remoção de arcos simétricos (A.2.2), portanto o par de caminhos de custo aditivo mínimo entre a e e, na rede da figura 2.1, é *abce* e *ade*. Esse par de caminhos é disjunto nos nós e tem custo aditivo $5+7=12$.

Caso não existisse o arco (a,d) o caminho mais curto entre a e e seria na mesma *abce*, enquanto que o caminho mais curto na rede modificada (figura 3.1) seria agora $ab'b''e$, ou seja, *abe*. Os caminhos *abce* e *abe* formariam um par de caminhos maximamente disjuntos que teriam o arco (a,b) em comum.

A diferença entre o algoritmo descrito em [1] e a versão aqui apresentada encontra-se no algoritmo da divisão dos nós (A.2.1), na condição que impõe que quando o arco simétrico de um arco do caminho mais curto do nó de origem para o nó de destino não existir deve ser criado. Sem esta imposição poderia não ser obtido nenhum par de caminhos ou poderia ser obtido um par de caminhos cujo custo aditivo não seria mínimo.

Capítulo 4

Algoritmos para a determinação de um par de caminhos maximamente disjuntos nos nós e nos SRLG

4.1 Introdução

Considere-se uma rede onde existem SRLG. Para garantir a proteção contra falhas das ligações deve garantir-se que o caminho ativo e o caminho de proteção constituem um par de caminhos disjuntos nos SRLG, isto é, os arcos pertencentes ao caminho ativo não deverão pertencer a nenhum dos SRLG que contenham arcos do caminho de proteção. O problema do cálculo de caminhos totalmente disjuntos nos nós e nos SRLG de custo aditivo mínimo foi já tratado em [13]. Quando não é possível obter pares de caminhos disjuntos nos SRLG deve tentar obter-se um par de caminhos maximamente disjuntos nos SRLG.

Neste capítulo pretende-se obter um par de caminhos maximamente disjuntos nos nós e nos SRLG, tal que o seu custo seja mínimo. Isto corresponde a resolver um problema multicritério, para o qual se definem as seguintes funções objetivo:

$$f_1(p,q) = |(V_p \cap V_q) \setminus \{s,t\}| \tag{4.1}$$

$$f_2(p,q) = |(A_p \cap A_q)| \tag{4.2}$$

$$f_3(p,q) = |R_p \cap R_q| \tag{4.3}$$

$$f_4(p,q) = |[(A_{R_p} \cup A_{R_q}) \cap (A_p \cup A_q)] \setminus (A_p \cap A_q)| \tag{4.4}$$

$$f_5(p,q) = d(p) + d(q) \tag{4.5}$$

sendo p e q caminhos de s para t. A função $f_1(p,q)$ devolve o número de nós intermédios

comuns a p e q; a função $f_2(p,q)$ devolve o número de arcos comuns a p e q; a função $f_3(p,q)$ devolve o número de SRLG comuns a p e q; a função $f_4(p,q)$ devolve o número de arcos de p ou q que podem falhar devido a um risco comum a p e q, excluindo os arcos comuns; e a função $f_5(p,q)$ devolve o custo aditivo do par de caminhos p e q.

Pode considerar-se uma abordagem lexicográfica para a ordenação das soluções deste problema, em que cada um dos problemas de otimização é resolvido sequencialmente [8]:

$$\min_{p,q \in \mathscr{P}_{st}} f_i(p,q) \tag{4.6}$$

sujeito a $p \neq q$ e $f_j(p,q) \leq f_j(p_j^*, q_j^*)$, $j = 1,2,\ldots i-1$ $(i > 1)$, $i = 1,2,3,4,5$, em que f_i é a i-ésima função objetivo e i representa a posição (ordem de importância) de uma função na sequência de preferência. O par de caminhos (p_j^*, q_j^*) representa o ótimo da j-ésima função obtida na j-ésima iteração.

Além da ordem indicada para as funções na formulação do problema (opção designada por 's') foi também considerada em alternativa a troca da posição relativa das funções (4.3) e (4.4) (opção essa que será designada por 'a').

Vão ser aqui apresentadas três heurísticas para a resolução deste problema: o Modified IMSH (MdIMSH), o Modified CoSE-MS (MdCoSE-MS) e o Modified Trap Avoidance (MdTA)[1]. Nenhuma destas heurísticas garante a obtenção de uma solução ou que a solução obtida é ótima, exceto eventualmente no par obtido na primeira iteração. A primeira iteração nas três heurísticas consiste no cálculo de um par de caminhos maximamente disjuntos nos nós de custo aditivo mínimo. Caso esse par de caminhos seja totalmente disjunto nos SRLG, o algoritmo termina, pois esse par de caminhos é a solução ótima.

As três heurísticas têm ainda em comum a determinação de caminhos semente, ou seja, do caminho a partir do qual se procura obter uma solução para o problema da equação (4.6), diferindo entre si na forma como são determinadas as redes onde esses caminhos são calculados.

A heurística MdIMSH que se baseia no algoritmo IMSH (*Iterative Modified Suurballe's Heuristic*) descrito em [15], recorre a um algoritmo de enumeração de caminhos mais curtos a partir dos quais procura obter um par de caminhos maximamente disjuntos nos SRLG. A heurística Modified CoSE-MS (MdCoSE-MS) baseia-se no algoritmo Modified CoSE (*Conflicting SRLG Exclusion*) descrito em [11] e no CoSE-MS descrito em [5], e procura obter o caminho mais curto em redes onde foram removidos os arcos

[1] As heurísticas MdIMSH e MdCoSE-MS foram propostas pela Professora Teresa Gomes. A heurística MdTA foi proposta pelo autor desta dissertação com base na experiência adquirida no estudo e implementação das duas heurísticas anteriores.

pertencentes a um dado conjunto de SRLG. Finalmente a heurística Modified Trap Avoidance (MdTA), que se baseia no algoritmo *Trap Avoidance* (TA) descrito em [17], obtém o caminho semente numa rede na qual vai sendo sucessivamente removido um arco.

Este capítulo apresenta na secção 4.2 o algoritmo da Heurística MdIMSH, na secção 4.3 o algoritmo da heurística MdCoSE-MS e finalmente na secção 4.4 o algoritmo da heurística MdTA.

4.2 Algoritmo da heurística MdIMSH (Modified Iterative Modified Suurballe's Heuristic)

A primeira iteração do algoritmo MdIMSH[2] consiste no cálculo de um par de caminhos maximamente disjuntos nos nós de custo aditivo mínimo, como já foi referido. Se o par obtido não for disjunto nos SRLG, será necessário então enumerar os caminhos entre o nó de origem e o nó de destino, por ordem não decrescente do seu custo, procurando melhorar a qualidade deste primeiro par. Para esse efeito usou-se o algoritmo MPS[3] [10] ou em alternativa o algoritmo de Yen[3] [18]. O algoritmo MPS é mais rápido que o algoritmo de Yen quando o número de caminhos a obter é elevado [9], mas para um número reduzido de iterações não era certo que o MPS conduzisse a um menor tempo de CPU. Por este motivo foram recolhidos tempos de execução do MdIMSH para estes dois algoritmos de enumeração de caminhos.

Após o cálculo do caminho semente é calculado um par de caminhos maximamente disjuntos nos nós e nos SRLG através do algoritmo *Modified Suurballe's Heuristic Extended* (MSHE) (como é explicado mais abaixo). Em cada iteração, o par de caminhos devolvido pelo MSHE passará a ser a nova solução (utilizando o método *mayUpdateSolution*) caso seja melhor que o par de caminhos anteriormente guardado, segundo um dos critérios definidos para a atualização de solução, opção 's' ou 'a' (ver secção anterior).

O algoritmo MdIMSH termina quando não existem mais caminhos semente entre o nó de origem e o de destino, ou quando é atingido o número máximo de iterações permitidas. Ao terminar devolve o melhor par de caminhos encontrado (caso tenha encontrado algum).

[2]No âmbito de [13] foram implementadas as heurísticas CoSE-MS e IMSH, tendo o autor desta dissertação reutilizado algumas sub-rotinas genéricas de manipulação de SRLG e implementado algumas adicionais.

[3]O estudo dos algoritmos MPS e Yen não foi considerado no âmbito deste trabalho. O código do algoritmo MPS e o código do algoritmo de Yen foram disponibilizados pela Professora Doutora Teresa Martinez Gomes.

Algoritmo 2: Algoritmo MdIMSH (Modified Iterative Modified Suurballe's Heuristic) para o cálculo de um par de caminhos maximamente disjuntos nos nós e nos SRLG

Dados: Grafo $G = (V,A)$, nó de origem s, nó de destino t, custo de cada arco l_k com $k \in A$, SRLG $R(i,j)$ associados a cada arco (i,j)

Resultado: Par de caminhos maximamente disjuntos nos nós e nos SRLG, entre s e t

1 $(p,q) \leftarrow \text{maxDisj}(s,t)$ //Cálculo de um par de caminhos maximamente disjuntos nos nós de custo aditivo mínimo, usando o algoritmo 1
2 //Seja R_p o conjunto dos SRLG de p e R_q o conjunto dos SRLG de q
3 **se** $R_p \cap R_q = \emptyset \wedge (p,q) \neq (\emptyset,\emptyset)$ **então**
4 | Termina //Termina com solução ótima se p e q não tiverem nenhum SRLG em comum
5 **fim**
6 $i \leftarrow 1$
7 **enquanto** $i < i_{\max}$ **faça**
8 | $p_s \leftarrow \text{kSP}(s,t)$ //Cálculo do i-ésimo caminho mais curto (semente) de s para t
9 | **se** $p_s = \emptyset$ **então**
10 | | Termina
11 | **fim**
12 | $(p',q') \leftarrow \text{MSHE}(s,t,p_s)$
13 | $(p,q) \leftarrow \text{mayUpdateSolution}(p',q',p,q)$
14 | $i \leftarrow i+1$
15 **fim**

O algoritmo MSHE, utilizado no MdIMSH, é uma extensão da *Modified Suurballe's Heuristic* (MSH) [15], a qual por sua vez é uma modificação do algoritmo de Suurballe [14]. A MSH é semelhante ao algoritmo de Bhandari para o cálculo de um par de caminhos maximamente disjuntos nos nós de custo aditivo mínimo (descrito no capítulo 3). Uma das diferenças do MSHE relativamente ao algoritmo de Bhandari é que, na divisão dos nós (algoritmo 4) é atribuído custo 0 (em vez de custo negativo) aos arcos simétricos dos arcos do caminho semente, o que evita que surjam ciclos negativos na rede, uma vez que no MdIMSH serão utilizados vários caminhos (semente) entre o nó de origem e o nó de destino e não apenas o caminho mais curto. Outra diferença é a soma da constante[4] x ao custo dos arcos que partilhem um ou mais SRLG com qualquer arco do caminho semente, por forma a tentar evitar tanto quanto possível, que o segundo caminho utilize algum SRLG do caminho semente.

Algoritmo 3: Algoritmo MSHE (Modified Suurballe's Heuristic Extended) para o cálculo de um par de caminhos maximamente disjuntos nos nós e nos SRLG

Dados: Grafo $G = (V,A)$, nó de origem s, nó de destino t, custo de cada arco l_k com $k \in A$, caminho semente p_s

Resultado: Par de caminhos maximamente disjuntos nos nós e nos SRLG, entre s e t

1 Transformar a rede (G) de acordo com o procedimento no algoritmo 4.
2 q' = Dijkstra(s,t)
3 Repor a rede (G) na sua forma original antes da transformação efetuada na linha 1.
4 $(p,q) \leftarrow$ removeInterlacing(p_s, q')

Na figura 4.1 encontra-se a representação de uma rede de exemplo com dois SRLG (um dos SRLG é constituído pelos arcos (a,b), (a,c), (b,a) e (c,a), enquanto o outro é constituído pelos arcos (f,h), (h,f) e (g,h)). Nessa figura os arcos do caminho mais curto entre a e i estão assinalados com traço mais espesso. Na figura 4.2 encontra-se a rede da figura 4.1 após a execução do algoritmo de divisão dos nós utilizado pelo MSHE (algoritmo 4). Com p_s (na linha 8 do algoritmo 2) dado por $acfhi$ (ver figura 4.1), o caminho mais curto q' (linha 2 do algoritmo 3) é dado por $adgf'c''beh'h''i$, ou seja, $adgfcbehi$, o qual possui o arco (f,c) simétrico ao arco (c,f) do caminho p_s. Após a remoção desses dois arcos (linha 4 do algoritmo 3) o caminho p_s e o caminho q' resultam no par de caminhos $(p,q) = (acbehi, adgfhi)$.

[4] Recorde-se, do capítulo 3, $x = (|V|-1)y$, com $y = L+\varepsilon$, $L = \sum_{k \in A} l_k$ e onde ε é um número positivo arbitrário.

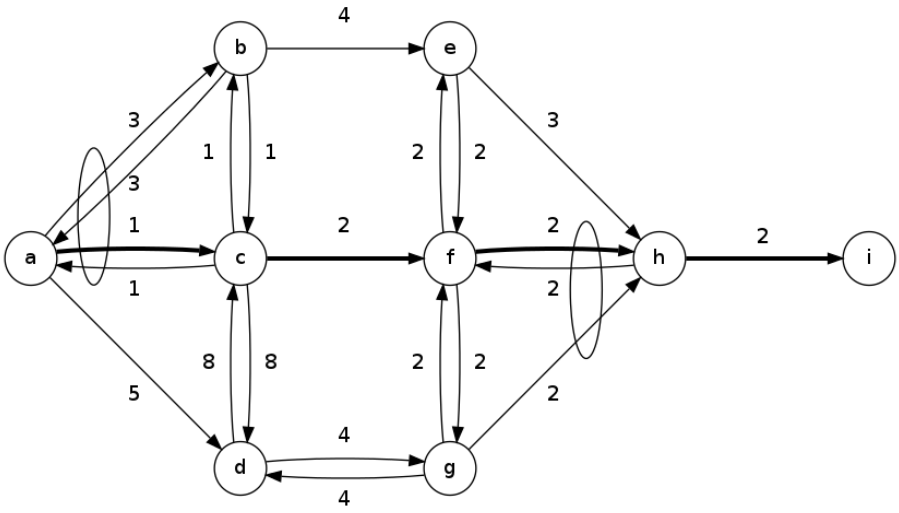

Figura 4.1: Rede de exemplo, com dois SRLG.

Algoritmo 4: Algoritmo de divisão dos nós utilizado pelo MSHE (doSplittingMSHE)

Dados: Grafo $G = (V,A)$, nó de origem s, nó de destino t, custo de cada arco l_k com $k \in A$, caminho semente p_s de s para t

Resultado: Grafo modificado $G' = (V',A')$

1 Criar o arco simétrico de cada arco do caminho semente, caso não exista.

2 Dividir cada nó do caminho semente (exceto o nó de origem e o de destino) em dois "sub-nós" (um de índice ' e outro de índice ") ligados entre si por dois arcos simétricos.

3 Ligar a cabeça de cada arco externo originalmente tendo como cabeça um nó do caminho semente, ao "sub-nó" de índice '. Ligar a cauda de cada arco externo originalmente tendo como cauda um nó do caminho semente, ao "sub-nó" de índice ".

4 Atribuir custo y aos arcos com direção do nó origem para o nó destino que ligam os "sub-nós" criados (ou seja, a cada arco que representa um nó no caminho semente p_s). Atribuir custo 0 aos arcos simétricos desses.

5 Atribuir custo $x + l_k$ aos arcos do caminho p_s, com custo original l_k. Aos seus arcos simétricos (que devem ser criados, se não existirem) atribuir custo 0.

6 Atribuir aos restantes arcos k que têm SRLG em comum com os arcos de p_s o custo $x + l_k$.

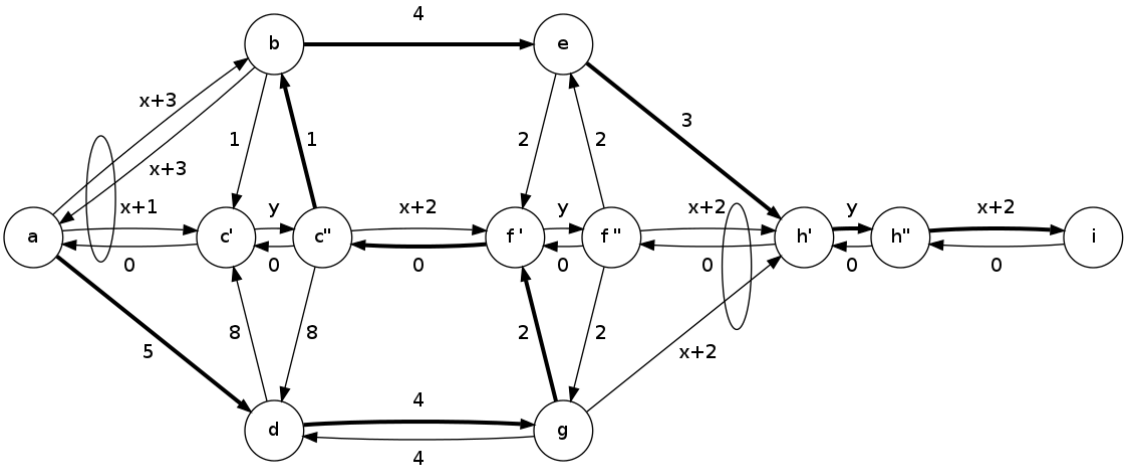

Figura 4.2: Rede da figura 4.1 após execução do algoritmo 4.

4.3 Algoritmo da heurística MdCoSE-MS (Modified Conflicting SRLG Exclusion - Min Sum)

4.3.1 Heurísticas auxiliares

A heurística MdCoSE-MS utiliza a heurística MSHE e ainda a extensão da MBH (*Modified Bhandari's Heuristic*), designada por MBHE (*Modified Bhandari's Heuristic Extended*). A extensão da MBH é semelhante à extensão da MSH, diferindo no algoritmo utilizado para a transformação da rede. O algoritmo da MBHE utiliza para esse efeito, na linha 1, o algoritmo auxiliar 18 no anexo A.2.3. Este último difere do procedimento de divisão dos nós traduzido pelo algoritmo 4 no custo atribuído aos arcos simétricos do caminho p_s, o qual toma o valor do simétrico do custo dos arcos correspondentes no caminho p_s.

Algoritmo 5: Algoritmo MBHE (Modified Bhandari's Heuristic Extended) para o cálculo de um par de caminhos maximamente disjuntos nos nós e nos SRLG

Dados: Grafo $G = (V,A)$, nó de origem s, nó de destino t, custo de cada arco l_k com $k \in A$, caminho semente p_s

Resultado: Par de caminhos maximamente disjuntos nos nós e nos SRLG, de s para t

1 Transformar a rede (G) de acordo com o algoritmo 18 (anexo A.2.3)
2 $q' = \text{Dijkstra}(s,t)$
3 Repor a rede (G) na sua forma original antes da transformação efetuada na linha 1
4 $(p,q) \leftarrow \text{removeInterlacing}(p_s, q')$

A heurística MBHE tende a obter caminhos mais curtos do que a heurística MSHE, a razão pela qual é utilizada no MdCoSE-MS. No entanto, a MBHE só pode ser usada quando o caminho semente foi calculado na rede original.

4.3.2 Algoritmo MdCoSE-MS

Tal como no MdIMSH, em primeiro lugar é calculado um par de caminhos maximamente disjuntos nos nós de custo aditivo mínimo, que caso seja totalmente disjunto nos SRLG corresponde à melhor solução possível, não sendo necessário continuar a execução do algoritmo.

Sejam I (o conjunto de inclusão) e E (o conjunto de exclusão) dois subconjuntos disjuntos de R (conjunto dos SRLG na rede). Designe-se por $P(I,E,H)$ o problema de calcular um caminho ativo e o correspondente caminho de proteção, usando como caminho semente o caminho mais curto numa rede em que os arcos afetados pelos SRLG no conjunto $E \cup H$ foram removidos, e onde H é a união de todos os conjuntos de exclusão que deram origem ao problema corrente P. Os elementos do conjunto de inclusão I não podem ser excluídos em problemas que resultem da divisão do problema P.

O caminho semente é usado pelas heurísticas MBHE ou MSHE para obter um par de caminhos maximamente disjuntos. Se esse par existe mas não é disjunto nos SRLG, novos problemas poderão ser gerados calculando o que se designa por SRLG em conflito.

Seja K o conjunto dos SRLG críticos, ou seja, o conjunto dos SRLG tal que nenhum caminho de s para t pode ser encontrado no grafo $G^r = (V, A \backslash A_r)$, para todo o $A_r \in K$. Estes SRLG não poderão nunca pertencer ao conjunto E (e por conseguinte ao conjunto H) pois nesse caso não será possível obter um caminho semente. Designe-se por K_V o conjunto dos nós críticos, ou seja, dos nós intermédios comuns ao par de caminhos maximamente disjuntos nos nós entre s e t.

O MdCoSE-MS utiliza uma pilha de problemas, onde são inseridos os novos problemas gerados e de onde são removidos os problemas resolvidos (com ou sem solução). Após o problema inicial (problema com todos os seus conjuntos associados vazios, $P_0(I,E,H) = P(\emptyset,\emptyset,\emptyset)$) ser inserido na pilha de problemas, é calculado um caminho semente, é determinado o conjunto dos SRLG críticos e é determinado o número de nós críticos. Após estes passos o algoritmo prossegue com as iterações. Em cada iteração faz:

- Obter o problema que se encontra no topo da pilha de problemas (problema corrente). Criar a rede onde vai ser calculado o caminho semente, removendo os arcos pertencentes aos SRLG contidos nos conjuntos E e H. Retirar o problema corrente da pilha de problemas.

- Calcular o caminho mais curto (caminho semente) na rede modificada.
- Repor o estado inicial da rede.
- Executar o algoritmo MBHE caso o problema corrente seja o problema inicial ou o algoritmo MSHE caso contrário. O algoritmo MBHE só pode ser utilizado com o caminho mais curto da rede original, pois caso fosse utilizado com qualquer outro caminho poderia dar origem a ciclos negativos.
- Atualizar a solução, caso seja melhor que a anteriormente guardada, segundo a opção 's' ou opção 'a' (ver secção 4.1).
- Caso a solução obtida não seja disjunta nos SRLG, determinar o conjunto de SRLG em conflito T relativo ao problema corrente. Se os nós intermédios comuns coincidem com os nós críticos, o conjunto T é dado pelos SRLG do caminho semente que são comuns ao par de caminhos obtido (excluindo os SRLG críticos); caso contrário T é obtido usando o algoritmo 8. Em seguida é necessário dividir o problema corrente em $|T|$ problemas e inseri-los na pilha de problemas.

  Seja o problema corrente $P_c(I_c, E_c, H_c)$, e $T = \{r_1, r_2, \ldots, r_{|T|}\}$, com $H = E_c \cup H_c$. Serão gerados e colocados na pilha os seguintes problemas: $P(I_c, \{r_1\}, H)$, $P(I_c \cup \{r_1\}, \{r_2\}, H)$, $P(I_c \cup \{r_1, r_2\}, \{r_3\}, H)$, $\ldots$, $P(I_c \cup \{r_1, r_2 \ldots, r_{|T|-1}\}, \{r_{|T|}\}, H)$.

O algoritmo termina assim que a pilha de problemas ficar vazia ou quando for atingido o número máximo de problemas permitidos.

O algoritmo 8 consiste em, dados o caminho semente p_s associado ao problema corrente e o conjunto de inclusão I_c associado ao problema corrente, retirar da rede os arcos pertencentes a cada SRLG de p_s que não pertença ao conjunto I_c nem ao conjunto K e calcular o caminho mais curto de s para t na rede resultante, até que não exista nenhum caminho de s para t ou sejam retirados da rede todos os SRLG de p_s que sejam comuns aos caminhos que forem sendo calculados. Cada SRLG removido é adicionado ao conjunto T.

Algoritmo 6: Algoritmo MdCoSE-MS (Modified Conflicting SRLG Exclusion - Min Sum) para o cálculo de um par de caminhos maximamente disjuntos nos nós e nos SRLG

Dados: Grafo $G = (V,A)$, nó de origem s, nó de destino t, custo de cada arco l_k com $k \in A$, SRLG $R(i,j)$ associados a cada arco (i,j)

Resultado: Par de caminhos maximamente disjuntos nos nós e nos SRLG, de s para t

$(p,q) \leftarrow$ maxDisj(s,t) //Cálculo de um par de caminhos maximamente disjuntos nos nós de custo aditivo mínimo
se $R_p \cap R_q = \emptyset \wedge (p,q) \neq (\emptyset,\emptyset)$ **então** Termina //Termina com solução ótima
$P_0 \leftarrow (\emptyset,\emptyset,\emptyset)$ //P_0 é o problema inicial.
//Seja S a pilha de problemas (inicialmente vazia)
Push(S,P_0) //Insere o problema inicial no topo da pilha S
$i \leftarrow 1$
$p_s \leftarrow$ Dijkstra(s,t) //Calcula o caminho mais curto
$K \leftarrow$ criticalSRLGs(s,t,p_s) //(Algoritmo 7)
$N_{K_V} \leftarrow |(V_p \cap V_q) \backslash \{s,t\}|$ //Nº de nós críticos
enquanto $\neg$empty$(S) \wedge i < i_{\max}$ **faça**
    $P_c(I_c,E_c,H_c) \leftarrow$ Top(S) //Problema corrente
    Pop(S)
    removeSRLGsArcs$(E_c \cup H_c)$ //Remove da rede os SRLG de E_c e de H_c
    $p_s \leftarrow$ Dijkstra(s,t) //Calcula caminho mais curto para o problema corrente
    undoRemoveSRLGsArcs$(E_c \cup H_c)$ //Repõe os SRLG de E_c e de H_c
    se $p_s = \emptyset$ **então** Termina
    se $(I_c,E_c,H_c) = (\emptyset,\emptyset,\emptyset)$ **então** $(p',q') \leftarrow$ MBHE(s,t,p_s)
    senão $(p',q') \leftarrow$ MSHE(s,t,p_s)
    $(p,q) \leftarrow$ mayUpdateSolution(p',q',p,q)
    se $(R_{p'} \cap R_{q'}) \backslash K \neq \emptyset$ **então**
        se $N_{K_V} = |(V_{p'} \cap V_{q'}) \backslash \{s,t\}|$ **então** $T \leftarrow R_{p_s} \cap ((R_{p'} \cap R_{q'}) \backslash K)$ **senão**
            $T \leftarrow$ SRLGExclusion(s,t,p_s,I_c,K)
        fim
        se $T \neq \emptyset$ **então**
            //Seja T o conjunto $\{r_1,r_2,\ldots,r_{|T|}\}$
            $P_1(I_1,E_1,H_1) \leftarrow P(I_c,r_1,E_c \cup H_c)$
            Push(P_1)
            $j \leftarrow 2$
            enquanto $j \leq |T|$ **faça**
                $P_j(I_j,E_j,H_j) \leftarrow P(I_{j-1} \cup E_{j-1},r_j,E_c \cup H_c)$
                Push(P_j)
                $j \leftarrow j+1$
            fim
        fim
    fim
    $i \leftarrow i+1$
fim

Algoritmo 7: Algoritmo criticalSRLGs para determinação dos SRLG da rede que são críticos

Dados: Grafo $G = (V,A)$, nó de origem s, nó de destino t, custo de cada arco l_k com $k \in A$, SRLG $R(i,j)$ associados a cada arco (i,j), caminho mais curto p_s de s para t

Resultado: Conjunto dos SRLG críticos K

```
1  //Seja R_{p_s} = {r_1, r_2, ..., r_{|R_{p_s}|}} o conjunto dos SRLG de p_s
2  K ← ∅
3  i ← 1
4  enquanto i ≤ |R_{p_s}| faça
5      removeSRLGArcs(r_i) //Remove da rede os arcos pertencentes a r_i
6      p' ← BFS(s,t)
7      se p' = ∅ então
8          K ← K ∪ {r_i}
9      fim
10     undoRemoveSRLGArcs(r_i) //Repor a rede no estado anterior
11     i ← i + 1
12 fim
```

Algoritmo 8: Algoritmo SRLGExclusion para determinação do conjunto dos SRLG em conflito T

Dados: Grafo $G = (V,A)$, nó de origem s, nó de destino t, custo de cada arco l_k com $k \in A$, SRLG $R(i,j)$ associados a cada arco (i,j), caminho semente associado ao problema corrente p_s, conjunto de inclusão associado ao problema corrente I_c, conjunto dos SRLG críticos K

Resultado: Conjunto de SRLG em conflito T

```
1  //Seja R_{p_s} = {r_1, r_2, ..., r_{|R_{p_s}|}} o conjunto dos SRLG de p_s
2  T ← ∅
3  R_{p_s} ← R_{p_s} \ (I_c ∪ K)
4  enquanto ¬empty(R_{p_s}) faça
5      //Seja r_j um dos elementos de R_{p_s}
6      T ← T ∪ {r_j}
7      R_{p_s} ← R_{p_s} \ {r_j}
8      removeSRLGArcs(r_j) //Remove da rede os arcos pertencentes a r_j
9      p' ← BFS(s,t)
10     se p' ≠ ∅ então
11         //Seja R_{p'} o conjunto dos SRLG de p'
12         R_{p_s} ← R_{p_s} ∩ R_{p'}
13     fim
14     senão
15         Termina
16     fim
17 fim
```

4.4 Algoritmo da heurística MdTA (Modified Trap Avoidance)

O algoritmo TA (*Trap Avoidance*), proposto em [17], tem como objetivo resolver o problema de cálculo de um par de caminhos disjuntos nos SRLG formulado como um problema *min-min*.

Uma das principais desvantagens apresentadas por determinadas heurísticas de cálculo de um par de caminhos disjuntos nos SRLG, é a queda em "armadilhas" ("*traps*") evitáveis [17]. Considera-se que determinada heurística caiu numa "armadilha" evitável quando falhou a determinação de um par de caminhos disjuntos nos SRLG [17], entre um determinado par de nós, e tal par de caminhos existe. Por exemplo, na figura A.3 uma vez calculado o caminho mais curto, de a para f, se os seus arcos forem removidos não é possível obter um caminho disjunto com este – o algoritmo caiu numa "armadilha", quando existe uma solução visível na figura A.5.

O TA é um algoritmo iterativo, em que cada iteração consiste em remover um arco da rede (designado de arco mais arriscado), calcular um caminho ativo na rede resultante, e por fim calcular um caminho de proteção na rede original. O objetivo de remover o arco mais arriscado é tentar evitar a queda em "armadilhas". É sugerido pelos autores do TA que seja escolhido para arco mais arriscado, o arco do caminho ativo que pertence ao maior número de SRLG comuns ao caminho ativo e ao caminho de proteção correntes [17].

Para resolver o problema *min-sum* de cálculo de um par de caminhos maximamente disjuntos nos SRLG, foi desenvolvida no âmbito do presente trabalho uma versão modificada do algoritmo TA, a qual foi designada por MdTA (*Modified Trap Avoidance*).

A primeira iteração do algoritmo MdTA (Algoritmo 9), tal como nos algoritmos anteriores, consiste em calcular um par de caminhos maximamente disjuntos nos nós, de custo aditivo mínimo. Caso o par de caminhos calculado seja totalmente disjunto nos SRLG está encontrada a melhor solução possível e o algoritmo termina. Caso o par de caminhos encontrado na primeira iteração não seja totalmente disjunto nos SRLG, é determinado o conjunto dos arcos críticos K_A, que é o conjunto dos arcos que são comuns ao par de caminhos determinado na primeira iteração, antes de o algoritmo prosseguir com as iterações seguintes. Nas iterações seguintes, em cada iteração o MdTA faz:

- Seja A o conjunto dos arcos da rede original e A' o conjunto dos arcos de A exceto aqueles que já foram escolhidos para arco mais arriscado. Calcular o caminho mais curto de s para t, com todos os arcos que já tenham sido escolhidos para arco mais arriscado $(A \backslash A')$ removidos da rede.

- Calcular um par de caminhos maximamente disjuntos nos nós e nos SRLG, de s para t, utilizando o algoritmo MSHE. Esse par de caminhos é identificado por (p', q').

- Atualizar a solução guardada caso a devolvida pelo MSHE seja melhor, segundo um dos critérios já indicados no MdIMSH.

- Determinar o arco mais arriscado da rede (*mostRiskyActiveLink*) e retirá-lo do conjunto A'. O arco mais arriscado é o arco do caminho semente que pertence ao maior número de SRLG comuns a p' e a q'.

  Caso exista mais do que um arco nessas circunstâncias, foram previstas duas possibilidades de desempate:

  opção 'g': escolher entre eles o arco emergente do nó de maior grau;

  opção 'c': escolher se possível um arco que não seja comum aos dois caminhos.

O algoritmo termina quando não for encontrado um caminho semente ou quando for atingido o número máximo de iterações permitidas.

Algoritmo 9: Algoritmo MdTA (Modified Trap Avoidance) para o cálculo de um par de caminhos maximamente disjuntos nos nós e nos SRLG

Dados: Grafo $G = (V,A)$, nó de origem s, nó de destino t, custo de cada arco l_k com $k \in A$, SRLG $R(i,j)$ associados a cada arco (i,j)

Resultado: Par de caminhos maximamente disjuntos nos nós e nos SRLG, de s para t

1 $(p,q) \leftarrow$ maxDisj(s,t) //Cálculo de um par de caminhos maximamente disjuntos nos nós de custo aditivo mínimo
2 **se** $R_p \cap R_q = \emptyset \wedge (p,q) \neq (\emptyset,\emptyset)$ **então**
3 | Termina //Termina com solução ótima se p e q não tiverem nenhum SRLG em comum
4 **fim**
5 $K_A \leftarrow A_p \cap A_q$ //Arcos críticos
6 $A' \leftarrow A$ //Conjunto dos arcos da rede
7 $i \leftarrow 1$
8 **enquanto** $i < i_{\max}$ **faça**
9 | $p_s \leftarrow$ BFS(s,t)
10 | **se** $p_s = \emptyset$ **então**
11 | | Termina
12 | **fim**
13 | undoRemoveArcs$(A \backslash A')$
14 | $(p',q') \leftarrow$ MSHE(s,t,p_s)
15 | $(p,q) \leftarrow$ mayUpdateSolution(p',q',p,q)
16 | $L \leftarrow$ mostRiskyActiveLink$(p_s, R_{p'} \cap R_{q'}, K_A)$
17 | $A' \leftarrow A' - \{L\}$
18 | removeArcs$(A \backslash A')$
19 | $i \leftarrow i+1$
20 **fim**

Capítulo 5

Algoritmo para a determinação de um par de caminhos disjuntos nas avarias, de custo aditivo mínimo

5.1 Introdução

Numa rede constituída por arcos suscetíveis a falhas e por arcos que são resilientes é vantajoso determinar pares de caminhos que sejam disjuntos nas avarias, isto é, pares de caminhos que apenas possam partilhar arcos resilientes.

Em [19] é proposto um algoritmo que calcula um par de caminhos de custo aditivo mínimo, tal que os arcos resilientes podem ser partilhados (o seu custo só é contabilizado uma única vez para o custo do par de caminhos). Os autores provam a correção do algoritmo e que o mesmo obtém a solução ótima em tempo polinomial.

O algoritmo proposto em [19] consiste essencialmente em três passos e precisa da definição de dois conjuntos: o conjunto dos nós de origem U^+, constituído pelo nó s e pelos nós cabeça dos arcos resilientes, e o conjunto dos nós de destino U^-, constituído pelo nó t e pelos nós cauda dos arcos resilientes. Os nós s e t pertencem unicamente a U^+ e U^-, respetivamente.

Se for desejado considerar que na rede existem nós resilientes, este algoritmo também pode ser utilizado [19], para obter caminhos disjuntos (nos nós) e nas avarias, desde que cada nó seja substituído por um arco que o represente. Para tal, cada nó i será representado por dois nós i' e i'' ligados por um arco (i',i''), tal que todos os arcos incidentes em i passam a ser incidentes em i' e todos os arcos emergentes de i passam a ser emergentes do nó i''. Se o nó for resiliente, então o arco (i',i'') será um arco resiliente. Nesta rede, o algoritmo proposto em [19] obtém o par de caminhos de custo aditivo mínimo, que apenas

Algoritmo 10: Algoritmo para cálculo de um par de caminhos disjuntos nas avarias, de custo aditivo mínimo

Dados: Grafo $G = (V,A)$, nó de origem s, nó de destino t, custo de cada arco l_k com $k \in A$, e $\bar{S} \subset A$ o conjunto dos arcos resilientes

Resultado: Par de caminhos disjuntos nas avarias, de custo aditivo mínimo, de s para t

1 Determinar um par de caminhos disjuntos *nos arcos* de cada nó de origem u ($u \in U^+$) para cada um dos nós de destino v ($v \in U^-$); construir uma versão modificada da rede, eliminando todos os arcos não resilientes e substituindo cada par de caminhos disjuntos nos arcos obtido por um arco com cauda igual ao nó de origem do par, cabeça igual ao nó de destino do par e custo igual ao custo do par. Caso a rede modificada contenha arcos paralelos será incluído apenas um desses arcos (o de menor custo, e para custo igual o arco resiliente) de forma a poder ser utilizado o algoritmo de Dijkstra ou BFS.

2 Calcular o caminho mais curto na rede construída no passo anterior.

3 Expandir o caminho calculado, substituindo cada arco pelo par de caminhos disjuntos correspondente (determinado no passo 1), e obter o par de caminhos disjuntos de custo mínimo nessa rede.

podem partilhar nós resilientes e arcos resilientes cujos extremos sejam também eles nós resilientes.

5.2 Novo problema

Considere-se agora o seguinte problema: *numa rede em que apenas existe informação acerca de arcos resilientes, deseja-se que o par de caminhos obtidos possa partilhar um ou mais desses arcos resilientes, tal que o seu custo seja mínimo e o par de caminhos seja totalmente disjunto nos nós ou formado por pares de "sub-caminhos" disjuntos nos nós ligados por arcos resilientes (cujo custo só é considerado uma única vez).*

Numa primeira aproximação a este novo problema, considerou-se a utilização do algoritmo em [19], em que os conjuntos U^+ e U^- são definidos como no caso em que se pretende considerar apenas falhas nos arcos não resilientes, mas em que os pares de caminhos obtidos dos elementos em U^+ para os elemento em U^-, são caminhos disjuntos nos nós. Desta forma apenas os nós em $U^+ \cup U^-$ poderão surgir como nós em comum na solução obtida, sem que um arco resiliente com extremo nesse nó seja partilhado pelo par.

Segue-se a descrição do algoritmo com esta primeira abordagem (algoritmo 11). Seja o nó de origem do par de caminhos pretendido designado por s e o respetivo nó de destino designado por t.

Algoritmo 11: Primeira abordagem ao cálculo de um par de caminhos disjuntos nos nós, de custo aditivo mínimo, numa rede com arcos resilientes

Dados: Grafo $G = (V,A)$, nó de origem s, nó de destino t, custo de cada arco l_k com $k \in A$, e $\bar{S} \subset A$ o conjunto dos arcos resilientes

Resultado: Par de caminhos disjuntos nas avarias, de custo aditivo mínimo, de s para t

1 Determinar um par de caminhos disjuntos *nos nós* de cada nó de origem u ($u \in U^+$) para cada um dos nós de destino v ($v \in U^-$), utilizando o algoritmo de Bhandari [1] para cálculo de um par de caminhos disjuntos nos nós de custo aditivo mínimo (implementado no âmbito do trabalho [13]).

2 Construir uma versão modificada da rede, eliminando todos os arcos não resilientes e substituindo cada par de caminhos disjuntos nos nós obtido por um arco com cauda igual ao nó de origem do par, cabeça igual ao nó de destino do par e custo igual ao custo do par. Caso a rede modificada contenha arcos paralelos será incluído apenas um desses arcos (o de menor custo, e para custo igual o arco resiliente) de forma a poder ser utilizado o algoritmo de Dijkstra ou BFS [1].

3 Determinar o caminho mais curto na rede modificada obtida no passo anterior, usando o algoritmo de Dijkstra ou o BFS [1].

4 Construir a versão reduzida da rede, substituindo cada um dos arcos da rede modificada que pertencem ao caminho mais curto determinado no passo 3, pelo seu respetivo par de caminhos disjuntos nos nós determinado no passo 1.

5 Determinar um par de caminhos maximamente disjuntos nos nós (não resilientes) na rede reduzida (que podem partilhar arcos resilientes e os nós em $U^+ \cup U^-$).

Na implementação do algoritmo 11 deve ter-se em atenção o seguinte:

- No passo 1, caso existam nós que façam simultaneamente parte do conjunto U^+ e do conjunto U^-, não é calculado o par de caminhos disjuntos nos nós de um nó para si mesmo.

- No passo 2 a rede modificada pode conter pares de arcos paralelos (arcos que têm a mesma cauda e a mesma cabeça), constituídos por um arco resiliente e um arco equivalente a um par de caminhos disjuntos nos nós determinado no passo 1. Não é desejável a existência de arcos paralelos, pois os algoritmos para cálculo do caminho mais curto implementados podem falhar na sua presença. Sendo assim, no passo 2, um dos arcos paralelos de cada par de arcos paralelos é removido segundo o seguinte critério:

  - Caso o custo do arco resiliente seja igual ou inferior ao custo do arco equivalente a um par de caminhos disjuntos nos nós, este último é eliminado.
  - Caso contrário o arco resiliente é eliminado.

Os algoritmos 10 e 11, embora obtenham soluções em tempo polinomial, requerem a resolução de $|U^-| \times |U^+| - |U^+ \cap U^-|$ problemas de cálculo de pares de caminhos disjuntos nos arcos e nos nós, respetivamente.

No sentido de garantir que o par de caminhos obtido não tem em comum nenhum nó pertencente a $U^+ \cap U^-$, sem que um arco resiliente com extremo nesse nó seja partilhado pelo par de caminhos, é proposta uma heurística[1] traduzida pelo algoritmo 12, que procura reduzir o número de chamadas ao algoritmo 11, mas não garante a obtenção de um par de custo mínimo. Uma vez que os nós comuns (que não são extremos de arcos resilientes partilhados), que surgem na solução obtida pelo algoritmo 11 pertencem a U^+ ou a U^-, a remoção de arcos protegidos com extremo no primeiro desses nós, permite em alguns casos obter uma solução. Foi assim proposta a heurística traduzida pelo algoritmo 12, a qual termina assim que encontra uma solução admissível numa dada iteração.

Na figura 5.1 encontra-se representada uma rede com dois arcos resilientes, (d,e) e (e,d). Nessa rede existe um par de caminhos disjuntos nos nós e nas avarias do nó a para o nó f, o par de caminhos $abdef$ e $acdehf$ de custo $6+4=10$, que é a solução ótima. Os dois caminhos têm um arco em comum, que é resiliente, e naturalmente o nó cauda e o nó cabeça desse arco. O algoritmo 11 não devolve essa solução, em vez disso devolve o par de caminhos $abdf$ e $acdgef$ de custo $4+5=9$, que não é disjunto nos nós e nas avarias (de acordo com a formulação apresentada nesta secção), uma vez que ambos os

[1] Esta heurística foi proposta pela Professora Doutora Teresa Martinez Gomes.

Algoritmo 12: Heurística para o cálculo de um par de caminhos disjuntos nos nós, de custo aditivo mínimo, numa rede com arcos resilientes

Dados: Grafo $G = (V,A)$, nó de origem s, nó de destino t, custo de cada arco l_k com $k \in A$, e $\bar{S} \subset A$ o conjunto dos arcos resilientes

Resultado: Par de caminhos disjuntos nas avarias, de s para t

1 Executar o algoritmo 11, para obter o par de caminhos p e q.
2 **se** $p = \emptyset \vee q = \emptyset$ **então** Termina (sem solução)
3 //A função foundSolution(p,q) devolve verdade se os caminhos são totalmente disjuntos ou se os nós intermédios comuns são extremos de arcos resilientes partilhados.
4 **se** *foundSolution*(p,q) **então** Termina (com solução)
5 $stop \leftarrow$ false
6 $stop1 \leftarrow$ false
7 $stop2 \leftarrow$ false
8 **Enquanto** $\neg stop$ **fazer**
9 Inativar arcos protegidos cuja cauda seja o primeiro nó comum a p e q.
10 Executar o algoritmo 11 para obter o par de caminhos p_1 e q_1.
11 Repõe a rede no estado anterior à transformação da linha 9.
12 Inativar arcos protegidos cuja cabeça seja o primeiro nó comum a p e q.
13 Executar o algoritmo 11 para obter o par de caminhos p_2 e q_2.
14 Repõe a rede no estado anterior à transformação da linha 12.
15 **se** $p_1 \neq \emptyset \wedge q_1 \neq \emptyset \wedge p_2 = q_2 = \emptyset$ **então**
16 $p \leftarrow p_1, q \leftarrow q_1$
17 $stop1 \leftarrow$ foundSolution(p,q)
18 **fim**
19 **se** $p_1 = q_1 = \emptyset \wedge p_2 \neq \emptyset \wedge q_2 \neq \emptyset$ **então**
20 $p \leftarrow p_2, q \leftarrow q_2$
21 $stop2 \leftarrow$ foundSolution(p,q)
22 **fim**
23 **se** *(stop1* $\wedge$ *stop2)* $\vee \neg$ *(stop1* $\vee$ *stop2))* **então**
24 **se** $p_1 \neq \emptyset \wedge q_1 \neq \emptyset \wedge p_2 \neq \emptyset \wedge q_2 \neq \emptyset$ **então**
25 //escolhe o par de menor custo para procurar solução
26 **se** $d(p_1,q_1) < d(p_2,q_2)$ **então**
27 $p \leftarrow p_1, q \leftarrow q_1$
28 **fim**
29 **senão**
30 $p \leftarrow p_2, q \leftarrow q_2$
31 **fim**
32 **fim**
33 **fim**
34 $stop \leftarrow stop1 \vee stop2$
35 **fim**
36 Repõe a rede no estado inicial.

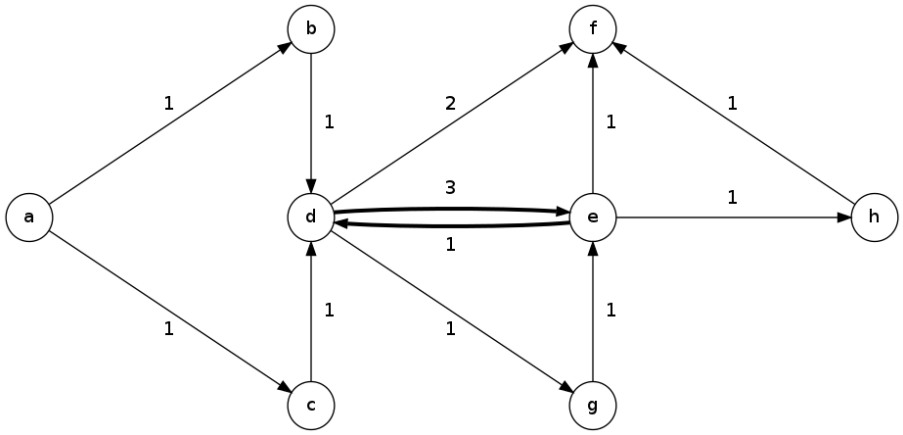

Figura 5.1: Exemplo de uma rede com dois arcos resilientes.

caminhos partilham o nó d e não partilham o arco resiliente que tem como cauda esse nó (o arco (d,e)). A heurística (algoritmo 12) encontra a solução ótima $abdef$ e $acdehf$.

Capítulo 6

Resultados

6.1 Introdução

Como já foi referido, um dos objetivos do trabalho foi fazer uma implementação dos algoritmos o mais eficiente possível do ponto de vista computacional, por forma a tornar viável a sua integração num PCE com recursos limitados.

Neste capítulo são apresentados resultados de desempenho dos algoritmos implementados, mais concretamente, tempos médios de cálculo de pares de caminhos e quantidade máxima de memória utilizada, num PCE cedido pela PT Inovação. O PCE tem as seguintes características: CPU G2_LE *core clock* 330MHz e 128MB de RAM.

São também apresentados resultados de testes realizados num computador, com o intuito de verificar a qualidade das soluções devolvidas por cada um dos três algoritmos de cálculo de um par de caminhos maximamente disjuntos nos nós e nos SRLG, por forma a obter uma base de comparação entre as soluções devolvidas por cada um. O computador utilizado tem as seguintes características: Intel® Core™ i7 CPU 870 @ 2.93GHz e 4GB de RAM.

Nas tabelas de resultados que se seguem são apresentados valores com um intervalo de confiança de 95%.

6.2 Algoritmo para a determinação de um par de caminhos maximamente disjuntos nos nós, de custo aditivo mínimo

A rede utilizada para a realização dos testes ao algoritmo para a determinação de um par de caminhos maximamente disjuntos nos nós, corresponde ao maior componente

Algoritmo	Tempo (na placa)	Tempo (no computador)
Bhandari (Dijkstra + Dijkstra Modificado)	$6,12 \pm 0,02$	$0,22 \pm 0,00$
Bhandari (Dijkstra + BFS)	$7,92 \pm 0,09$	$0,27 \pm 0,00$
Bhandari (BFS + BFS)	$8,18 \pm 0,11$	$0,25 \pm 0,00$

Tabela 6.1: Tempos (em segundos) de cálculo de 1000 pares de caminhos maximamente disjuntos nos nós, na placa da PT Inovação e num computador.

conectado de uma rede fornecida pela PT Inovação, com 361 nós e 511 arestas (representadas por 1022 arcos dirigidos).

Para testar o tempo de execução do algoritmo foi desenvolvido um programa, que consiste em escolher 1000 pares de nós origem-destino aleatoriamente e posteriormente contar o tempo despendido a calcular os 1000 pares de caminhos utilizando cada uma das três variantes do algoritmo. O programa repete o procedimento anterior dez vezes, sendo em cada uma delas os 1000 pares de nós origem-destino escolhidos aleatoriamente (usando sementes diferentes para o gerador de sequências pseudo-aleatórias).

Foram obtidos assim três conjuntos (um para cada variante do algoritmo 1) de dez resultados correspondentes ao tempo despendido a calcular 1000 pares de caminhos. Na tabela 6.1 encontram-se os valores médios dos tempos obtidos. Estes tempos de CPU indicam que esta rotina é perfeitamente adequada para utilização num PCE.

Observando a tabela 6.1 verifica-se que, na placa, a variante do algoritmo que utiliza o algoritmo de Dijkstra para o cálculo do caminho mais curto e o algoritmo de Dijkstra modificado para o cálculo do segundo caminho, foi cerca de 20% mais rápida do que as outras duas variantes.

Quanto ao consumo de memória, verificou-se que cada uma das variantes utilizou no máximo 4908 kiB de RAM durante a sua execução.

6.3 Algoritmos para a determinação de um par de caminhos maximamente disjuntos nos nós e nos SRLG

Para avaliação do desempenho dos algoritmos para a determinação de um par de caminhos maximamente disjuntos nos nós e nos SRLG, na placa cedida pela PT Inovação, foram geradas dez redes a partir da rede fornecida, com SRLG distribuídos aleatoriamente em cada uma delas. Em todos os testes realizados foi utilizada a opção 's'. A escolha da opção 's' ou 'a' traduz uma preferência do decisor e não é relevante para a análise relativa do desempenho das heurísticas.

O programa de testes desenvolvido consiste no cálculo de 1000 pares de caminhos, por cada uma das variantes dos três algoritmos implementados, em cada uma das dez redes de teste, utilizando também sementes diferentes para a escolha aleatória dos 1000 pares de nós origem-destino.

Os valores médios dos tempos obtidos encontram-se nas tabelas 6.3 e 6.4, nas quais os valores de $i_{\max}$ representam o número máximo de iterações/problemas que foi permitido efetuar. Como se pode observar as variantes do MdTA e o MdCoSE-MS são bastante mais rápidos do que as variantes do MdIMSH para um número máximo de iterações mais elevado, pois realizam um menor número de iterações (como se pode ver na tabela 6.6). Analisando as variantes do MdIMSH verifica-se que a que utiliza o algoritmo MPS para a enumeração de caminhos é cerca de duas vezes mais rápida para $i_{max} \geq 50$. Do MdTA a variante que demorou menos tempo a realizar os cálculos foi a 'c', uma vez que a variante 'g' exige mais processamento. O tempo de cálculo manteve-se constante, para ambas as variantes do MdTA, a partir de $i_{max} = 100$.

Todas as variantes utilizaram 5044 kiB de RAM, no máximo.

Além dos testes de desempenho, em termos de CPU e memória usando a rede da PT Inovação, foram ainda realizados testes com o objetivo de comparar a qualidade relativa das soluções devolvidas por cada uma das variantes dos três algoritmos implementados, para a determinação de um par de caminhos maximamente disjuntos nos nós e nos SRLG. Na tabela 6.5 verifica-se que para $i_{max} = 10$ o MdIMSH e o MdTA apresentam melhores resultados, contudo, quanto ao número médio de SRLG em comum nos pares obtidos, os resultados são semelhantes (ver tabela 6.7).

Foram ainda consideradas três redes de teste não dirigidas [12], em que cada aresta tem um custo dado pelo custo do primeiro módulo que lhe está associado. Cada aresta é representada por um par de arcos dirigidos simétricos – ver tabela 6.2. A partir de cada uma dessas três redes foram geradas dez redes, cada uma com uma distribuição de SRLG diferente, distribuição essa que leva em consideração a proximidade geográfica dos extremos dos arcos [4].

Os testes foram realizados num computador, com um programa que consiste na execução de cada uma das variantes dos três algoritmos implementados, fazendo o cálculo de um par de caminhos maximamente disjuntos nos nós e nos SRLG para todos os pares de nós origem-destino da rede, em cada uma das redes geradas.

Como se pode observar nas tabelas 6.8, 6.9 e 6.10, o MdIMSH é o algoritmo que obteve melhores resultados, isto é, é o algoritmo que obteve uma percentagem mais elevada de soluções totalmente disjuntas nos nós e nos SRLG. A qualidade das soluções não totalmente disjuntas obtidas pelo MdIMSH também foi melhor, como indicado pelo número médio de SRLG em comum (tabelas 6.14, 6.15 e 6.16). O inconveniente é que o tempo

Rede	Nº de nós	Nº de arcos
nobel-eu	28	82
cost266	37	114
germany50	50	176

Tabela 6.2: Características das redes de teste [12] utilizadas para os algoritmos de cálculo de um par de caminhos maximamente disjuntos nos nós e nos SRLG.

despendido pelo MdIMSH na realização dos cálculos foi bastante superior ao dos outros algoritmos, pois efetuou um maior número de iterações, como pode ser verificado nas tabelas 6.11, 6.12 e 6.13.

O MdTA produziu melhores resultados que o MdCoSE-MS, tendo em conta que apresenta soluções de melhor qualidade e que apesar de utilizar mais tempo de CPU do que este último, esse incremento de tempo não é excessivo para $i_{max} \leq 10$.

É de realçar o facto de o MdCoSE-MS e o MdTA terem estabilizado a partir de $i_{max} = 10$, o que significa que atingiram o seu melhor resultado com $i_{max} \leq 10$, não sendo necessário efetuar mais iterações.

Rede da PT Inovação					
	MdIMSH		MdCoSE-MS	MdTA	
$i_{\max}$	MPS	Yen		'c'	'g'
5	$76,9 \pm 3,4$	$47,8 \pm 1,8$	$48,4 \pm 2,0$	$38,4 \pm 1,1$	$38,3 \pm 1,1$
10	$99,5 \pm 4,6$	$97,7 \pm 3,9$	$50,3 \pm 2,2$	$64,8 \pm 1,9$	$66,7 \pm 2,2$
20	$145,3 \pm 6,2$	$199,5 \pm 8,3$	$53,8 \pm 2,7$	$98,1 \pm 3,4$	$109,0 \pm 4,1$
50	$288,5 \pm 11,7$	$509,6 \pm 21,1$	$63,9 \pm 4,4$	$161,9 \pm 12,1$	$211,9 \pm 8,8$
100	$537,7 \pm 21,6$	$1032,9 \pm 42,7$	$80,6 \pm 7,4$	$193,0 \pm 20,0$	$283,1 \pm 13,4$
200	$1046,1 \pm 41,7$	$2091,9 \pm 82,5$	$114,0 \pm 13,6$	$194,0 \pm 20,3$	$284,2 \pm 13,5$
500	$2631,5 \pm 104,2$	$5264,8 \pm 207,6$	$213,8 \pm 32,1$	$194,0 \pm 20,3$	$284,2 \pm 13,5$
1000	$5364,3 \pm 213,4$	$10663,9 \pm 429,0$	$379,8 \pm 63,0$	$194,0 \pm 20,3$	$284,2 \pm 13,5$

Tabela 6.3: Tempos de CPU (em segundos) correspondentes ao cálculo de 1000 pares de caminhos maximamente disjuntos nos nós e nos SRLG, na placa da PT Inovação.

Rede da PT Inovação					
	MdIMSH		MdCoSE-MS	MdTA	
$i_{\max}$	MPS	Yen		'c'	'g'
5	$2,37 \pm 0,09$	$1,43 \pm 0,06$	$1,38 \pm 0,06$	$1,17 \pm 0,04$	$1,18 \pm 0,05$
10	$3,04 \pm 0,12$	$2,89 \pm 0,13$	$1,45 \pm 0,07$	$1,97 \pm 0,07$	$2,05 \pm 0,09$
20	$4,42 \pm 0,16$	$5,85 \pm 0,28$	$1,55 \pm 0,09$	$2,98 \pm 0,11$	$3,35 \pm 0,16$
50	$8,73 \pm 0,31$	$14,92 \pm 0,71$	$1,88 \pm 0,14$	$4,93 \pm 0,40$	$6,48 \pm 0,36$
100	$16,18 \pm 0,59$	$30,28 \pm 1,45$	$2,42 \pm 0,24$	$5,87 \pm 0,67$	$8,70 \pm 0,54$
200	$31,52 \pm 1,17$	$61,44 \pm 2,93$	$3,51 \pm 0,45$	$5,89 \pm 0,67$	$8,73 \pm 0,54$
500	$79,19 \pm 2,99$	$155,94 \pm 7,39$	$6,74 \pm 1,07$	$5,89 \pm 0,68$	$8,74 \pm 0,53$
1000	$161,16 \pm 5,97$	$315,87 \pm 14,98$	$12,13 \pm 2,09$	$5,89 \pm 0,68$	$8,77 \pm 0,50$

Tabela 6.4: Tempos de CPU (em segundos) correspondentes ao cálculo de 1000 pares de caminhos maximamente disjuntos nos nós e nos SRLG, no computador.

Rede da PT Inovação					
	MdIMSH		MdCoSE-MS	MdTA	
$i_{\max}$	MPS	Yen		'c'	'g'
5	$45,23 \pm 1,20$	$45,25 \pm 1,18$	$43,21 \pm 1,18$	$46,19 \pm 1,11$	$46,12 \pm 1,12$
10	$46,04 \pm 1,17$	$46,02 \pm 1,17$	$43,24 \pm 1,17$	$46,64 \pm 1,08$	$46,65 \pm 1,10$
20	$46,46 \pm 1,13$	$46,44 \pm 1,13$	$43,26 \pm 1,16$	$46,71 \pm 1,07$	$46,78 \pm 1,08$
50	$46,77 \pm 1,09$	$46,74 \pm 1,09$	$43,26 \pm 1,16$	$46,71 \pm 1,08$	$46,80 \pm 1,07$
100	$46,92 \pm 1,08$	$46,88 \pm 1,08$	$43,27 \pm 1,15$	$46,72 \pm 1,08$	$46,81 \pm 1,07$
200	$47,01 \pm 1,07$	$46,97 \pm 1,07$	$43,27 \pm 1,15$	$46,72 \pm 1,08$	$46,81 \pm 1,07$
500	$47,11 \pm 1,07$	$47,04 \pm 1,07$	$43,27 \pm 1,15$	$46,72 \pm 1,08$	$46,81 \pm 1,07$
1000	$47,18 \pm 1,06$	$47,07 \pm 1,06$	$43,27 \pm 1,15$	$46,72 \pm 1,08$	$46,81 \pm 1,07$

Tabela 6.5: Percentagem média de soluções disjuntas nos SRLG (obtida na rede da PT Inovação).

Rede da PT Inovação					
	MdIMSH		MdCoSE-MS	MdTA	
$i_{\max}$	MPS	Yen		'c'	'g'
5	$3,75 \pm 0,08$	$3,77 \pm 0,08$	$2,38 \pm 0,04$	$3,72 \pm 0,08$	$3,73 \pm 0,08$
10	$7,18 \pm 0,18$	$7,22 \pm 0,18$	$2,59 \pm 0,07$	$6,36 \pm 0,15$	$6,49 \pm 0,17$
20	$14,06 \pm 0,38$	$14,14 \pm 0,38$	$2,94 \pm 0,13$	$9,19 \pm 0,26$	$9,91 \pm 0,30$
50	$34,67 \pm 0,97$	$34,87 \pm 0,98$	$3,98 \pm 0,34$	$13,76 \pm 0,76$	$16,85 \pm 0,55$
100	$69,03 \pm 1,97$	$69,44 \pm 1,98$	$5,72 \pm 0,69$	$15,82 \pm 1,14$	$21,20 \pm 0,64$
200	$137,74 \pm 3,95$	$138,57 \pm 3,98$	$9,18 \pm 1,39$	$15,87 \pm 1,15$	$21,27 \pm 0,64$
500	$343,78 \pm 9,91$	$345,97 \pm 9,97$	$19,57 \pm 3,49$	$15,87 \pm 1,15$	$21,27 \pm 0,64$
1000	$686,94 \pm 19,85$	$691,63 \pm 19,96$	$36,87 \pm 7,00$	$15,87 \pm 1,15$	$21,27 \pm 0,64$

Tabela 6.6: Número médio de iterações efetuadas (obtido na rede da PT Inovação).

Rede da PT Inovação					
	MdIMSH		MdCoSE-MS	MdTA	
$i_{\max}$	MPS	Yen		'c'	'g'
5	$2,42 \pm 0,08$	$2,42 \pm 0,08$	$2,42 \pm 0,07$	$2,43 \pm 0,08$	$2,43 \pm 0,08$
10	$2,42 \pm 0,08$	$2,42 \pm 0,08$	$2,42 \pm 0,08$	$2,44 \pm 0,08$	$2,44 \pm 0,08$
20	$2,42 \pm 0,08$	$2,42 \pm 0,08$	$2,41 \pm 0,08$	$2,44 \pm 0,08$	$2,44 \pm 0,08$
50	$2,42 \pm 0,08$	$2,42 \pm 0,08$	$2,41 \pm 0,08$	$2,44 \pm 0,08$	$2,44 \pm 0,08$
100	$2,42 \pm 0,08$	$2,42 \pm 0,08$	$2,41 \pm 0,08$	$2,44 \pm 0,08$	$2,44 \pm 0,08$
200	$2,43 \pm 0,08$	$2,43 \pm 0,08$	$2,41 \pm 0,08$	$2,44 \pm 0,08$	$2,44 \pm 0,08$
500	$2,43 \pm 0,08$	$2,43 \pm 0,08$	$2,41 \pm 0,08$	$2,44 \pm 0,08$	$2,44 \pm 0,08$
1000	$2,43 \pm 0,08$	$2,43 \pm 0,08$	$2,41 \pm 0,08$	$2,44 \pm 0,08$	$2,44 \pm 0,08$

Tabela 6.7: Número médio de SRLG em comum, considerando apenas as soluções que tenham SRLG em comum (obtido na rede da PT Inovação).

Rede nobel-eu					
	MdIMSH		MdCoSE-MS	MdTA	
$i_{\max}$	MPS	Yen		'c'	'g'
5	$57,70 \pm 3,67$	$57,69 \pm 3,67$	$51,19 \pm 2,30$	$57,09 \pm 3,57$	$57,31 \pm 3,59$
10	$60,36 \pm 4,27$	$60,38 \pm 4,29$	$51,23 \pm 2,34$	$58,53 \pm 3,96$	$59,27 \pm 4,30$
20	$61,19 \pm 4,54$	$61,19 \pm 4,54$	$51,24 \pm 2,36$	$58,56 \pm 3,96$	$59,33 \pm 4,27$
50	$62,00 \pm 4,99$	$62,00 \pm 4,99$	$51,24 \pm 2,36$	$58,56 \pm 3,96$	$59,33 \pm 4,27$
100	$62,30 \pm 5,08$	$62,30 \pm 5,08$	$51,24 \pm 2,36$	$58,56 \pm 3,96$	$59,33 \pm 4,27$
200	$62,41 \pm 5,22$	$62,41 \pm 5,22$	$51,24 \pm 2,36$	$58,56 \pm 3,96$	$59,33 \pm 4,27$
500	$62,43 \pm 5,22$	$62,43 \pm 5,22$	$51,24 \pm 2,36$	$58,56 \pm 3,96$	$59,33 \pm 4,27$
1000	$62,43 \pm 5,22$	$62,43 \pm 5,22$	$51,24 \pm 2,36$	$58,56 \pm 3,96$	$59,33 \pm 4,27$

Tabela 6.8: Percentagem média de soluções totalmente disjuntas (obtida na rede nobel-eu).

Rede cost266					
	MdIMSH		MdCoSE-MS	MdTA	
$i_{\max}$	MPS	Yen		'c'	'g'
5	$69,92 \pm 3,11$	$69,92 \pm 3,11$	$62,03 \pm 2,93$	$70,92 \pm 3,52$	$70,92 \pm 3,34$
10	$74,41 \pm 3,59$	$74,41 \pm 3,59$	$62,33 \pm 3,11$	$74,33 \pm 4,06$	$74,85 \pm 3,98$
20	$76,80 \pm 3,81$	$76,80 \pm 3,81$	$62,36 \pm 3,14$	$74,78 \pm 4,19$	$75,57 \pm 4,23$
50	$79,10 \pm 4,11$	$79,10 \pm 4,11$	$62,36 \pm 3,14$	$74,78 \pm 4,19$	$75,57 \pm 4,23$
100	$80,24 \pm 4,48$	$80,24 \pm 4,48$	$62,36 \pm 3,14$	$74,78 \pm 4,19$	$75,57 \pm 4,23$
200	$81,10 \pm 4,88$	$81,10 \pm 4,88$	$62,36 \pm 3,14$	$74,78 \pm 4,19$	$75,57 \pm 4,23$
500	$81,64 \pm 5,21$	$81,64 \pm 5,21$	$62,36 \pm 3,14$	$74,78 \pm 4,19$	$75,57 \pm 4,23$
1000	$82,01 \pm 5,40$	$82,01 \pm 5,40$	$62,36 \pm 3,14$	$74,78 \pm 4,19$	$75,57 \pm 4,23$

Tabela 6.9: Percentagem média de soluções totalmente disjuntas (obtida na rede cost266).

Rede germany50					
	MdIMSH		MdCoSE-MS	MdTA	
$i_{\max}$	MPS	Yen		'c'	'g'
5	$91,10 \pm 3,04$	$91,23 \pm 3,05$	$82,71 \pm 3,52$	$91,69 \pm 3,14$	$91,84 \pm 3,06$
10	$93,42 \pm 3,09$	$93,36 \pm 3,07$	$82,75 \pm 3,51$	$93,00 \pm 3,19$	$93,35 \pm 3,10$
20	$94,11 \pm 3,10$	$94,11 \pm 3,08$	$82,75 \pm 3,50$	$93,20 \pm 3,21$	$93,58 \pm 3,13$
50	$94,50 \pm 3,10$	$94,49 \pm 3,10$	$82,75 \pm 3,50$	$93,20 \pm 3,21$	$93,58 \pm 3,14$
100	$94,59 \pm 3,12$	$94,58 \pm 3,12$	$82,75 \pm 3,50$	$93,20 \pm 3,21$	$93,58 \pm 3,14$
200	$94,64 \pm 3,12$	$94,64 \pm 3,12$	$82,75 \pm 3,50$	$93,20 \pm 3,21$	$93,58 \pm 3,14$
500	$94,67 \pm 3,13$	$94,67 \pm 3,13$	$82,75 \pm 3,50$	$93,20 \pm 3,21$	$93,58 \pm 3,14$
1000	$94,70 \pm 3,13$	$94,70 \pm 3,12$	$82,75 \pm 3,50$	$93,20 \pm 3,21$	$93,58 \pm 3,14$

Tabela 6.10: Percentagem média de soluções totalmente disjuntas (obtida na rede germany50).

Rede nobel-eu					
	MdIMSH		MdCoSE-MS	MdTA	
$i_{\max}$	MPS	Yen		'c'	'g'
5	$3,87 \pm 0,17$	$3,87 \pm 0,17$	$2,53 \pm 0,10$	$3,79 \pm 0,17$	$3,81 \pm 0,17$
10	$7,46 \pm 0,39$	$7,46 \pm 0,39$	$3,68 \pm 0,28$	$5,27 \pm 0,30$	$5,74 \pm 0,30$
20	$14,64 \pm 0,83$	$14,64 \pm 0,82$	$5,88 \pm 0,64$	$5,38 \pm 0,31$	$6,05 \pm 0,33$
50	$36,17 \pm 2,13$	$36,17 \pm 2,13$	$12,38 \pm 1,70$	$5,38 \pm 0,31$	$6,05 \pm 0,33$
100	$72,07 \pm 4,30$	$72,07 \pm 4,30$	$23,19 \pm 3,45$	$5,38 \pm 0,31$	$6,05 \pm 0,33$
200	$143,85 \pm 8,64$	$143,85 \pm 8,64$	$44,80 \pm 6,95$	$5,38 \pm 0,31$	$6,05 \pm 0,33$
500	$355,34 \pm 21,24$	$355,34 \pm 21,23$	$109,64 \pm 17,46$	$5,38 \pm 0,31$	$6,05 \pm 0,33$
1000	$650,41 \pm 37,87$	$650,41 \pm 37,85$	$217,71 \pm 34,97$	$5,38 \pm 0,31$	$6,05 \pm 0,33$

Tabela 6.11: Número médio de iterações efetuadas (obtido na rede nobel-eu).

Rede cost266					
	MdIMSH		MdCoSE-MS	MdTA	
$i_{\max}$	MPS	Yen		'c'	'g'
5	$3,86 \pm 0,14$	$3,86 \pm 0,14$	$2,45 \pm 0,07$	$3,83 \pm 0,14$	$3,83 \pm 0,14$
10	$7,44 \pm 0,32$	$7,44 \pm 0,32$	$3,45 \pm 0,20$	$5,90 \pm 0,28$	$6,44 \pm 0,32$
20	$14,59 \pm 0,67$	$14,59 \pm 0,67$	$5,40 \pm 0,49$	$6,31 \pm 0,38$	$7,62 \pm 0,47$
50	$36,04 \pm 1,72$	$36,04 \pm 1,72$	$11,15 \pm 1,36$	$6,31 \pm 0,38$	$7,62 \pm 0,47$
100	$71,79 \pm 3,47$	$71,79 \pm 3,47$	$20,70 \pm 2,76$	$6,31 \pm 0,38$	$7,62 \pm 0,47$
200	$143,29 \pm 6,98$	$143,29 \pm 6,98$	$39,77 \pm 5,57$	$6,31 \pm 0,38$	$7,62 \pm 0,47$
500	$357,79 \pm 17,49$	$357,79 \pm 17,49$	$96,98 \pm 13,97$	$6,31 \pm 0,38$	$7,62 \pm 0,47$
1000	$715,30 \pm 35,02$	$715,30 \pm 35,02$	$192,32 \pm 27,98$	$6,31 \pm 0,38$	$7,62 \pm 0,47$

Tabela 6.12: Número médio de iterações efetuadas (obtido na rede cost266).

Rede germany50					
	MdIMSH		MdCoSE-MS	MdTA	
$i_{\max}$	MPS	Yen		'c'	'g'
5	$3,42 \pm 0,16$	$3,42 \pm 0,16$	$1,82 \pm 0,09$	$3,39 \pm 0,16$	$3,40 \pm 0,16$
10	$6,44 \pm 0,37$	$6,44 \pm 0,36$	$2,06 \pm 0,15$	$5,63 \pm 0,31$	$5,96 \pm 0,32$
20	$12,48 \pm 0,78$	$12,49 \pm 0,76$	$2,52 \pm 0,27$	$7,31 \pm 0,43$	$9,88 \pm 0,53$
50	$30,60 \pm 2,02$	$30,62 \pm 1,95$	$3,91 \pm 0,64$	$7,35 \pm 0,44$	$11,63 \pm 0,63$
100	$60,80 \pm 4,07$	$60,85 \pm 3,95$	$6,22 \pm 1,26$	$7,35 \pm 0,44$	$11,63 \pm 0,63$
200	$121,20 \pm 8,19$	$121,30 \pm 7,94$	$10,84 \pm 2,48$	$7,35 \pm 0,44$	$11,63 \pm 0,63$
500	$302,40 \pm 20,53$	$302,66 \pm 19,90$	$24,69 \pm 6,16$	$7,35 \pm 0,44$	$11,63 \pm 0,63$
1000	$604,40 \pm 41,11$	$604,93 \pm 39,84$	$47,77 \pm 12,28$	$7,35 \pm 0,44$	$11,63 \pm 0,63$

Tabela 6.13: Número médio de iterações efetuadas (obtido na rede germany50).

Rede nobel-eu					
	MdIMSH		MdCoSE-MS	MdTA	
$i_{\max}$	MPS	Yen		'c'	'g'
5	$1,78 \pm 0,24$	$1,78 \pm 0,24$	$1,84 \pm 0,21$	$1,80 \pm 0,21$	$1,82 \pm 0,21$
10	$1,63 \pm 0,21$	$1,63 \pm 0,21$	$1,80 \pm 0,20$	$1,71 \pm 0,21$	$1,70 \pm 0,21$
20	$1,54 \pm 0,20$	$1,54 \pm 0,20$	$1,79 \pm 0,20$	$1,71 \pm 0,21$	$1,69 \pm 0,21$
50	$1,48 \pm 0,18$	$1,48 \pm 0,18$	$1,78 \pm 0,20$	$1,71 \pm 0,21$	$1,69 \pm 0,21$
100	$1,46 \pm 0,18$	$1,46 \pm 0,18$	$1,78 \pm 0,20$	$1,71 \pm 0,21$	$1,69 \pm 0,21$
200	$1,45 \pm 0,18$	$1,45 \pm 0,18$	$1,78 \pm 0,20$	$1,71 \pm 0,21$	$1,69 \pm 0,21$
500	$1,43 \pm 0,17$	$1,43 \pm 0,17$	$1,78 \pm 0,20$	$1,71 \pm 0,21$	$1,69 \pm 0,21$
1000	$1,42 \pm 0,17$	$1,42 \pm 0,17$	$1,78 \pm 0,20$	$1,71 \pm 0,21$	$1,69 \pm 0,21$

Tabela 6.14: Número médio de SRLG em comum, considerando apenas as soluções que tenham SRLG em comum (obtido na rede nobel-eu).

Rede cost266					
	MdIMSH		MdCoSE-MS	MdTA	
$i_{\max}$	MPS	Yen		'c'	'g'
5	$1,70 \pm 0,17$	$1,70 \pm 0,17$	$1,64 \pm 0,11$	$1,64 \pm 0,13$	$1,64 \pm 0,13$
10	$1,49 \pm 0,17$	$1,49 \pm 0,17$	$1,58 \pm 0,11$	$1,50 \pm 0,15$	$1,49 \pm 0,16$
20	$1,37 \pm 0,16$	$1,37 \pm 0,16$	$1,56 \pm 0,11$	$1,49 \pm 0,16$	$1,47 \pm 0,17$
50	$1,29 \pm 0,16$	$1,29 \pm 0,16$	$1,56 \pm 0,11$	$1,49 \pm 0,16$	$1,47 \pm 0,17$
100	$1,25 \pm 0,15$	$1,25 \pm 0,15$	$1,56 \pm 0,11$	$1,49 \pm 0,16$	$1,47 \pm 0,17$
200	$1,22 \pm 0,14$	$1,22 \pm 0,14$	$1,56 \pm 0,11$	$1,49 \pm 0,16$	$1,47 \pm 0,17$
500	$1,20 \pm 0,14$	$1,20 \pm 0,14$	$1,56 \pm 0,11$	$1,49 \pm 0,16$	$1,47 \pm 0,17$
1000	$1,20 \pm 0,14$	$1,20 \pm 0,14$	$1,56 \pm 0,11$	$1,49 \pm 0,16$	$1,47 \pm 0,17$

Tabela 6.15: Número médio de SRLG em comum, considerando apenas as soluções que tenham SRLG em comum (obtido na rede cost266).

Rede germany50					
	MdIMSH		MdCoSE-MS	MdTA	
$i_{\max}$	MPS	Yen		'c'	'g'
5	$1,15 \pm 0,07$	$1,13 \pm 0,05$	$1,26 \pm 0,05$	$1,16 \pm 0,09$	$1,16 \pm 0,09$
10	$1,06 \pm 0,03$	$1,06 \pm 0,03$	$1,24 \pm 0,05$	$1,14 \pm 0,07$	$1,12 \pm 0,07$
20	$1,03 \pm 0,03$	$1,03 \pm 0,02$	$1,24 \pm 0,05$	$1,14 \pm 0,07$	$1,11 \pm 0,07$
50	$1,02 \pm 0,02$	$1,02 \pm 0,02$	$1,24 \pm 0,05$	$1,14 \pm 0,07$	$1,12 \pm 0,07$
100	$1,02 \pm 0,02$	$1,02 \pm 0,02$	$1,24 \pm 0,05$	$1,14 \pm 0,07$	$1,12 \pm 0,07$
200	$1,01 \pm 0,01$	$1,01 \pm 0,01$	$1,24 \pm 0,05$	$1,14 \pm 0,07$	$1,12 \pm 0,07$
500	$1,01 \pm 0,01$	$1,01 \pm 0,01$	$1,24 \pm 0,05$	$1,14 \pm 0,07$	$1,12 \pm 0,07$
1000	$1,01 \pm 0,01$	$1,01 \pm 0,01$	$1,24 \pm 0,05$	$1,14 \pm 0,07$	$1,12 \pm 0,07$

Tabela 6.16: Número médio de SRLG em comum, considerando apenas as soluções que tenham SRLG em comum (obtido na rede germany50).

6.4 Algoritmo para a determinação de um par de caminhos disjuntos nas avarias, de custo aditivo mínimo

Para realizar os testes de desempenho do algoritmo para a determinação de um par de caminhos disjuntos nas avarias foram geradas dez redes, a partir da rede fornecida pela PT Inovação. Em cada uma das redes geradas 1% das suas arestas foram, de forma aleatória, tornadas protegidas.

O programa utilizado para realização dos testes é semelhante ao utilizado para os algoritmos anteriores. Consiste em calcular 1000 pares de caminhos, com cada uma das três variantes do algoritmo e cada uma das três variantes da heurística, em cada uma das 10 redes geradas.

Os valores médios dos tempos obtidos encontram-se na tabela 6.17. Verifica-se que para a mesma variante o algoritmo 11 apresenta melhores resultados do que a heurística (traduzida pelo algoritmo 12), embora a diferença seja pequena, uma vez que a heurística raramente invoca o algoritmo 11. No entanto, ao contrário do caso do algoritmo para a determinação de um par de caminhos maximamente disjuntos nos nós, a variante que utiliza o algoritmo de Dijkstra no cálculo do caminho mais curto e o algoritmo BFS no cálculo do segundo caminho, demorou em média mais tempo a realizar os cálculos do que a variante que utiliza o algoritmo BFS para calcular ambos os caminhos.

O valor máximo de RAM utilizada por cada uma das variantes do algoritmo e da heurística foi 5020 kiB.

Algoritmo	Variante	Tempo (na placa)	Tempo (no computador)
Algoritmo 11	Bhandari (Dij. + Dij. Mod.)	$1994,90 \pm 210,40$	$57,34 \pm 5,73$
Algoritmo 11	Bhandari (Dij. + BFS)	$2069,31 \pm 206,18$	$57,62 \pm 5,52$
Algoritmo 11	Bhandari (BFS + BFS)	$2048,14 \pm 210,50$	$53,59 \pm 5,69$
Algoritmo 12 (heurística)	Bhandari (Dij. + Dij. Mod.)	$2011,91 \pm 217,69$	$57,56 \pm 5,81$
Algoritmo 12 (heurística)	Bhandari (Dij. + BFS)	$2084,47 \pm 213,82$	$57,87 \pm 5,57$
Algoritmo 12 (heurística)	Bhandari (BFS + BFS)	$2058,84 \pm 216,39$	$53,97 \pm 5,75$

Tabela 6.17: Tempos (em segundos) de cálculo de 1000 pares de caminhos disjuntos nas avarias, na placa da PT Inovação e num computador.

Capítulo 7

Conclusão

7.1 Trabalho realizado

A garantia de serviços de comunicação, de forma ininterrupta e com elevado débito, requer a implementação de mecanismos de proteção. Essa proteção pode ser conseguida utilizando dois caminhos totalmente disjuntos para cada conexão, ou caso não seja possível, dois caminhos maximamente disjuntos. Neste contexto, o conceito de SRLG permite a uma camada superior da rede obter caminhos disjuntos nas falhas da camada inferior, sem que os detalhes desta sejam comunicados à primeira. Por outro lado, a informação acerca da existência de arcos resilientes (ou seja arcos protegidos numa camada inferior) permite a determinação de caminhos disjuntos nas avarias.

Esta dissertação foi desenvolvida no âmbito de um contrato de investigação e desenvolvimento em colaboração com a PT Inovação. As rotinas implementadas serão incorporadas na pilha protocolar a instalar num PCE. Por esse motivo foi necessário ter em especial atenção a utilização de memória e o tempo de execução das rotinas implementadas. Foi analisado o desempenho das rotinas implementadas num PCE (UNICOM-V5) disponibilizado pela PT Inovação.

Foi estudado e implementado um algoritmo para a determinação de pares de caminhos maximamente disjuntos nos nós, de custo aditivo mínimo. Verificou-se que o seu tempo de execução, para a rede de teste fornecida pela PT Inovação, era adequado à sua utilização no PCE fornecido.

Foram propostas e implementadas três heurísticas (MdIMSH, MdCoSE-MS e MdTA) para a determinação de pares de caminhos maximamente disjuntos nos nós e nos SRLG, procurando também minimizar o seu custo aditivo. As heurísticas MdIMSH, MdCoSE-MS [4] são extensões de heurísticas que procuram resolver o problema *min-sum* considerando SRLG. A heurística MdTA é uma modificação e extensão da heurística TA que

procura resolver o problema *min-min* considerando SRLG.

Foi implementada uma heurística que procura obter um par de caminhos disjuntos nos nós (exceto nós extremos de arcos resilientes partilhados) e nas avarias, de custo aditivo mínimo, que utiliza uma adaptação do algoritmo proposto em [19].

Tendo em conta os resultados dos testes de desempenho efetuados, o algoritmo para o cálculo de um par de caminhos maximamente disjuntos nos nós, de custo aditivo mínimo, é perfeitamente adequado para a utilização num PCE com recursos limitados.

A determinação de caminhos maximamente disjuntos nos nós e nos SRLG é um problema difícil, contudo as heurísticas implementadas, utilizando apenas 10 iterações, conseguem obter bons resultados, num tempo de CPU ainda adequado à sua utilização no plano de controlo. A heurística MdCoSE-MS é a que apresenta pior desempenho do ponto de vista da qualidade das soluções obtidas. Contudo os valores para a percentagem de soluções disjuntas nos SRLG ou para o número médio de SRLG em comum são bastante próximos para as 3 heurísticas (ou seja, para o mesmo número de iterações há quase sempre alguma sobreposição dos intervalos de confiança em torno dos valores médios). Embora a heurística MdIMSH obtenha soluções ligeiramente melhores, requer mais tempo de CPU do que as outras heurísticas sempre que o número de iterações permitido é igual ou superior a 20. Os resultados obtidos indicam que a heurística MdTA será um bom compromisso entre eficiência computacional e qualidade das soluções obtidas.

A heurística para obtenção de um par de caminhos disjuntos nos nós e nas avarias, nos testes efetuados utilizando a rede fornecida pela PT Inovação, mostrou não ser adequada ao plano de controlo. O algoritmo utilizado na heurística, tal como o algoritmo proposto em [19], necessita de resolver um número elevado de problemas de pares de caminhos disjuntos nos nós de custo aditivo mínimo. Contudo o tempo de execução da heurística proposta é compatível com a sua utilização no plano de gestão.

7.2 Trabalho futuro

O estudo do desempenho das três heurísticas propostas para a determinação de pares de caminhos maximamente disjuntos nos nós e nos SRLG, deverá ter como referência a solução ótima. Para tal será necessário escrever a formalização do problema traduzido pela equação (4.6) e sua resolução usando o CPLEX [7].

A heurística para obtenção de um par de caminhos disjuntos nos nós (exceto nós extremos de arcos resilientes partilhados) e nas avarias, nem sempre obtém solução ou esta não é de custo mínimo. Permanece assim em aberto, a obtenção de um algoritmo ou heurística para a resolução deste problema.

Anexo A

Algoritmos auxiliares

A.1 Algoritmos para determinação do caminho mais curto entre dois nós de uma rede

Nos algoritmos que se seguem[1] o nó de origem é designado por s e o nó de destino por t. A restante notação segue a indicada na secção 2.5.

A.1.1 Algoritmo de Dijkstra

O algoritmo de Dijkstra determina o caminho mais curto entre dois nós de uma rede. Seja o custo do caminho do nó s para o nó i ($i \in V$) designado por $d(i)$ (soma do custo dos arcos de s para i) e o nó predecessor do nó i designado por $P(i)$. Relembra-se que Γ_i^+ é o conjunto dos nós que são cabeça de um arco cuja cauda é i e $l(i,j)$ é o custo do arco do nó i para o nó j. O algoritmo de Dijkstra pode ser apresentado na forma indicada no Algoritmo 13.

Após o algoritmo ter terminado, o caminho mais curto do nó s para o nó t é obtido lendo os nós predecessores, começando do nó t.

O valor infinito atribuído a $d(i)$, no algoritmo, tem de ser superior ao custo do caminho mais curto a ser determinado. Como esse custo não é conhecido à partida, pode considerar-se o pior cenário possível, que é existir um caminho mais curto de custo igual à soma de todos os arcos da rede ($\sum l_k$). O valor infinito pode então na prática ser igual a $\sum l_k$.

O algoritmo de Dijkstra pode falhar na determinação da solução correta, quando aplicado em redes que contenham arcos de custo negativo.

[1] O código dos algoritmos de cálculo do caminho mais curto e do algoritmo de remoção de arcos simétricos foi implementado pelo autor de [13].

Algoritmo 13: Algoritmo de Dijkstra

Dados: Grafo $G = (V,A)$, nó de origem s, nó de destino t, custo de cada arco l_k com $k \in A$

Resultado: Caminho mais curto de s para t

$d(s) \leftarrow 0$
$d(i) \leftarrow l(s,i)$, para todos os nós $i \in \Gamma_s^+$
$d(i) \leftarrow \infty$, para os restantes nós do grafo
$P(i) \leftarrow s$, $\forall\, i \in V$
$S \leftarrow V - \{s\}$
$j \leftarrow s$
Enquanto $j \neq t$ **fazer**
    Procurar $j \in S$, tal que $d(j) \leftarrow \min\, d(i), i \in S$
    $S \leftarrow S - \{j\}$
    $\forall\, i \in \Gamma_j$ e $i \in S$, **se** $d(j) + l(j,i) < d(i)$ **então**
        $d(i) \leftarrow d(j) + l(j,i)$
        $P(i) \leftarrow j$
    fim
fim

A.1.2 Algoritmo de Dijkstra modificado

Algoritmo 14: Algoritmo de Dijkstra modificado

Dados: Grafo $G = (V,A)$, nó de origem s, nó de destino t, custo de cada arco l_k com $k \in A$

Resultado: Caminho mais curto de s para t

$d(s) \leftarrow 0$
$d(i) \leftarrow l(s,i)$, para todo os nós $i \in \Gamma_s^+$
$d(i) \leftarrow \infty$, para os restantes nós do grafo
$P(i) \leftarrow s$, $\forall\, i \in V$
$S \leftarrow V - \{s\}$
$j \leftarrow s$
Enquanto $j \neq t$ **fazer**
    Procurar $j \in S$, tal que $d(j) \leftarrow \min\, d(i), i \in S$
    $S \leftarrow S - \{j\}$
    $\forall\, i \in \Gamma_j$, **se** $d(j) + l(j,i) < d(i)$ **então**
        $d(i) \leftarrow d(j) + l(j,i)$
        $P(i) \leftarrow j$
        $S \leftarrow S \cup \{i\}$
    fim
fim

O algoritmo de Dijkstra modificado é uma variante do algoritmo de Dijkstra, que permite a determinação correta do caminho mais curto em redes que contenham arcos de custo negativo (mas não ciclos negativos).

Após a execução do algoritmo, obtém-se a sequência de nós que formam o caminho mais curto de s para t, lendo os nós predecessores, começando do nó t.

A.1.3 Algoritmo BFS (Breadth-First Search)

É uma alternativa ao algoritmo de Dijkstra modificado, que pode também ser aplicado a redes com arcos de custo negativo. Seja Γ^T o conjunto de nós a partir dos quais a pesquisa é feita numa dada iteração e Γ^I o conjunto de nós cujas etiquetas são atualizadas nessa iteração.

Algoritmo 15: Algoritmo BFS (Breadth-First Search)

Dados: Grafo $G = (V,A)$, nó de origem s, nó de destino t, custo de cada arco l_k com $k \in A$

Resultado: Caminho mais curto de s para t

```
1  d(s) ← 0
2  d(i) ← ∞, ∀ i ∈ V (i ≠ s)
3  P(i) ← s, ∀ i ∈ V
4  Γ^T ← {s}
5  Enquanto Γ^T ≠ ∅ fazer
6  |  Γ^I ← ∅
7  |  Para todo j ∈ Γ^T fazer
8  |  |  Para todo i ∈ Γ_j^+ fazer
9  |  |  |  se d(j) + l(j,i) < d(i) ∧ d(j) + l(j,i) < d(t) então
10 |  |  |  |  d(i) ← d(j) + l(j,i)
11 |  |  |  |  P(i) ← j
12 |  |  |  |  Γ^I ← Γ^I \ {i}
13 |  |  |  fim
14 |  |  fim
15 |  fim
16 |  Γ^I ← Γ^I \ {t}
17 |  Γ^T ← Γ^I
18 fim
```

Após a execução do algoritmo, obtém-se a sequência de nós que formam o caminho mais curto de s para t, lendo os nós predecessores, começando do nó t.

Se existirem dois ou mais caminhos mais curtos, entre s e t, o algoritmo BFS devolve aquele que tiver menor número de arcos.

A.2 Outros algoritmos auxiliares

A.2.1 Algoritmo de divisão dos nós (*node-splitting*), utilizado pelo algoritmo 1

O algoritmo 16 realiza a modificação à rede original, necessária para o cálculo do segundo caminho mais curto, no algoritmo de Bhandari para o cálculo de um par de caminhos maximamente disjuntos nos nós.

Algoritmo 16: Algoritmo de divisão dos nós, utilizado pelo algoritmo 1

Dados: Grafo $G = (V,A)$, nó de origem s, nó de destino t, custo de cada arco l_k com $k \in A$, caminho mais curto de s para t
Resultado: Grafo modificado $G' = (V',A')$

1 Criar o arco simétrico de cada arco do caminho mais curto, caso não exista.
2 Dividir cada nó do caminho mais curto (exceto o nó de origem e o de destino) em dois "sub-nós" ligados por um arco e pelo arco simétrico desse. Ligar a cabeça de cada arco externo originalmente tendo como cabeça um nó do caminho mais curto, ao "sub-nó" de índice '. Ligar a cauda de cada arco externo originalmente tendo como cauda um nó do caminho mais curto, ao "sub-nó" de índice ".
3 Atribuir custo y aos arcos que ligam um "sub-nó" de índice ' a um "sub-nó" de índice ". Atribuir custo 0 aos arcos simétricos desses.
4 Atribuir custo $x + l_k$ aos arcos do caminho mais curto, com custo original l_k. Aos seus arcos simétricos, caso existam (se não existirem, criá-los), atribuir custo $-l_k$.

No passo 1 deste algoritmo não é necessário criar o arco simétrico do primeiro arco, nem o arco simétrico do último arco, do caminho mais curto:

- Não é necessário criar o arco simétrico do primeiro arco do caminho mais curto, uma vez que o nó de origem de um caminho nunca pode ter um nó predecessor, caso contrário o algoritmo estaria em ciclo.

- Não é necessário criar o arco simétrico do último arco do caminho mais curto, pois os algoritmos para cálculo do caminho mais curto, já descritos, terminam assim que a pesquisa chega ao nó de destino. Por esse motivo, nenhum arco que tem como cauda o nó de destino é usado para melhoria do custo do caminho entre o nó de origem e qualquer outro nó da rede.

As figuras A.1 e A.2 mostram um exemplo de aplicação deste algoritmo, em que se considera que $acfhi$ é o caminho mais curto (os arcos que constituem o caminho mais curto estão representados com traço mais espesso).

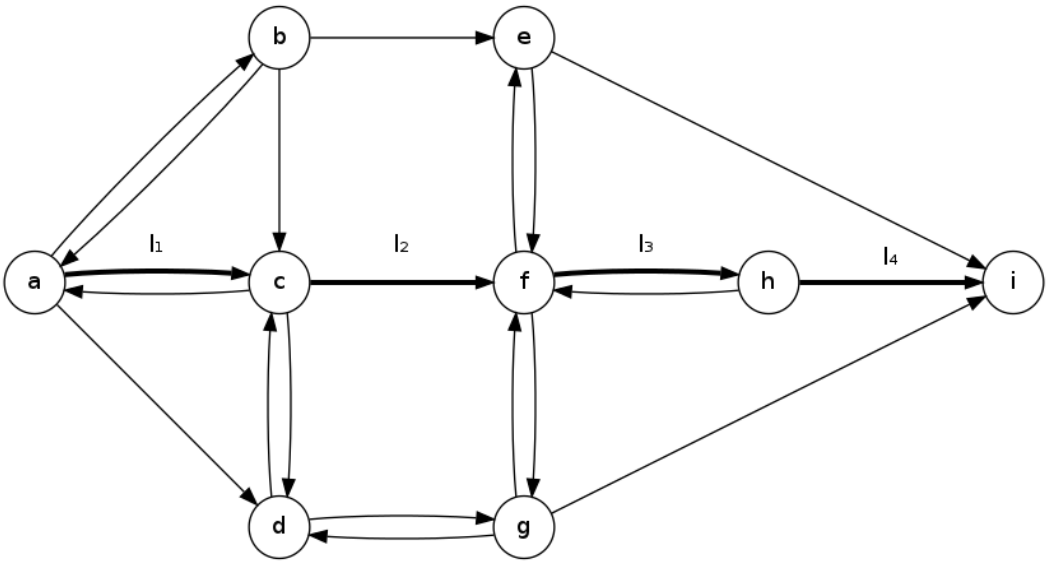

Figura A.1: Exemplo de uma rede.

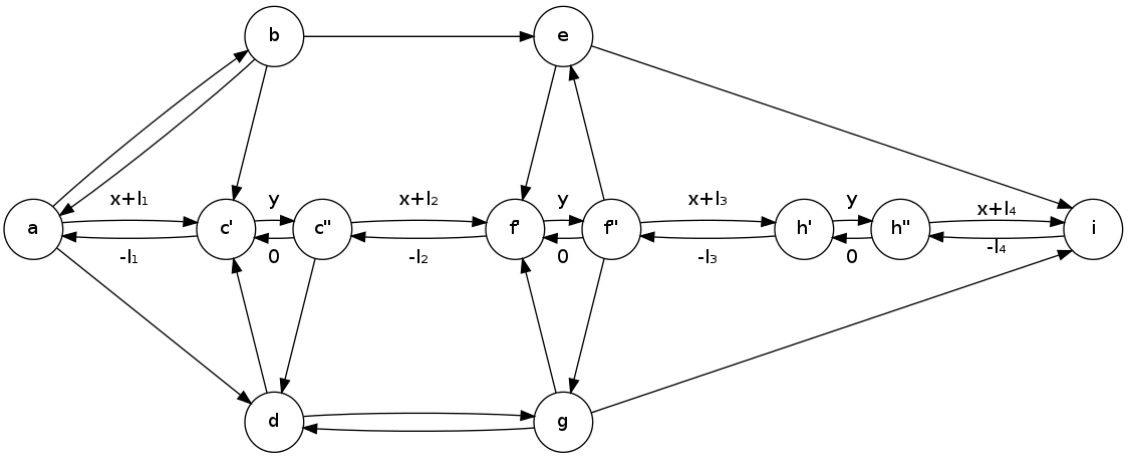

Figura A.2: Rede da figura anterior após divisão dos nós.

A.2.2 Algoritmo de remoção de arcos simétricos (*remove-interlacing*)

O algoritmo apresentado de seguida permite remover os arcos simétricos, pertencentes a um par de caminhos.

Algoritmo 17: Algoritmo de remoção de arcos simétricos

Dados: Um par de caminhos (p,q), o nó de origem s, o nó de destino t
Resultado: Um par de caminhos maximamente disjuntos nos nós, de s para t

1 Para cada segmento $(m,n) \in p$, **se** $(n,m) \in q$ **então**
2 $\quad p = \{p^1,(m,n),p^2\}; q = \{q^1,(n,m),q^2\}$
3 $\quad p \leftarrow \{q^1,p^2\}$
4 $\quad q \leftarrow \{p^1,q^2\}$
5 **fim**

A figura A.3 mostra um exemplo de aplicação deste algoritmo. Considerando que o caminho mais curto do nó a para o nó f na rede representada na figura seguinte é $abdf$ e que o segundo caminho mais curto é $acdbef$, por aplicação deste algoritmo os arcos bd e db são removidos, dando origem ao par de caminhos disjuntos nos nós $acdf$ e $abef$.

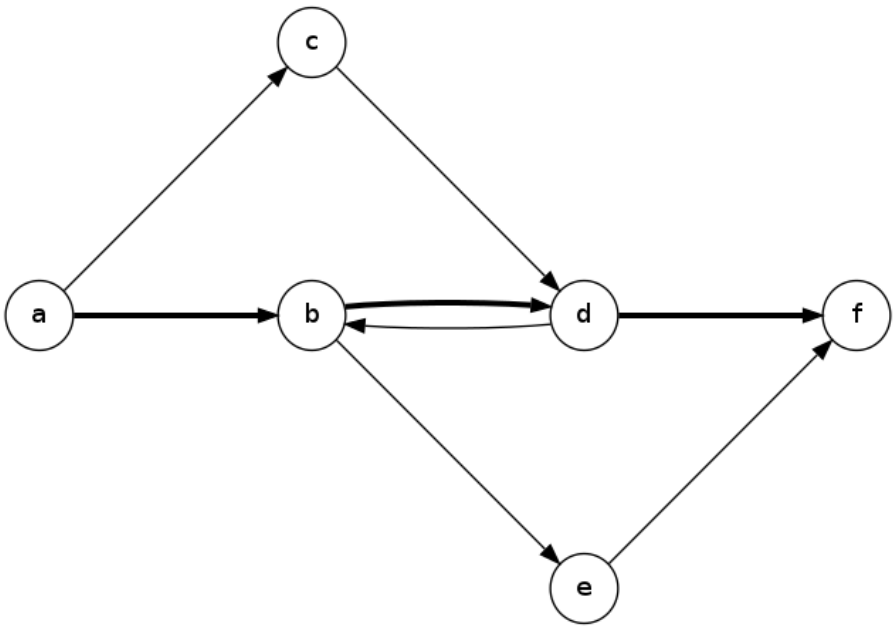

Figura A.3: Rede de exemplo com os arcos do caminho mais curto realçados.

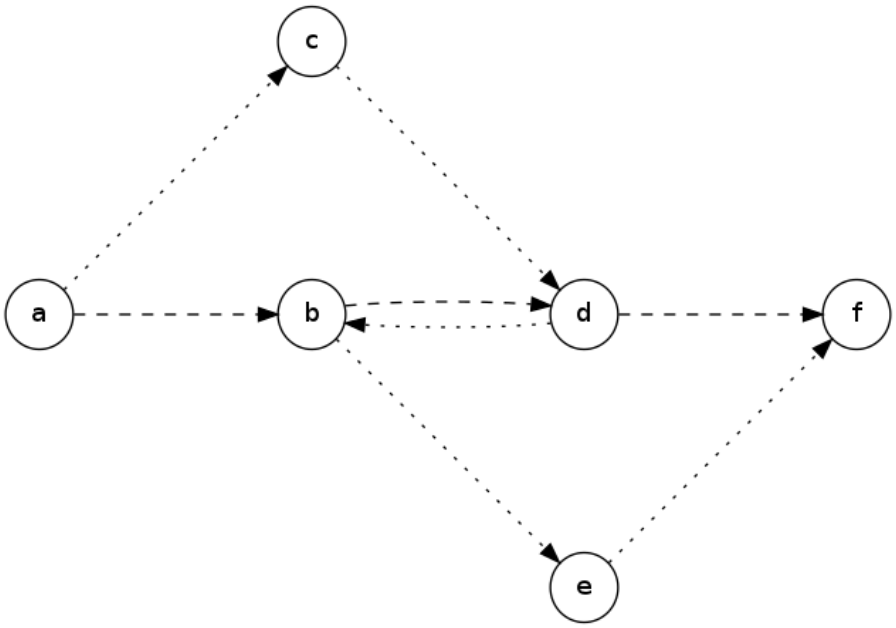

Figura A.4: Par de caminhos antes da remoção de arcos simétricos.

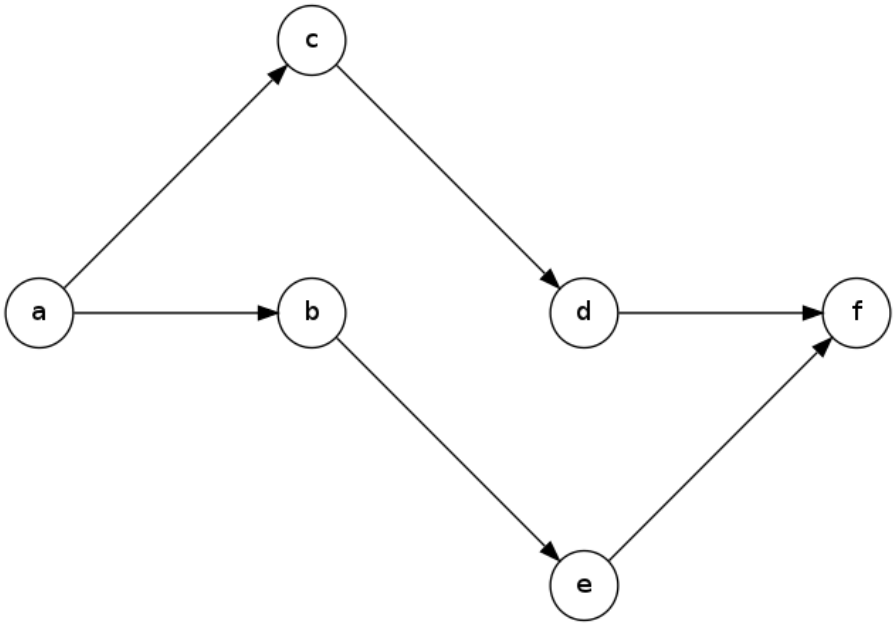

Figura A.5: Par de caminhos disjuntos nos nós, obtido através da remoção de arcos simétricos.

A.2.3 Algoritmo de divisão dos nós da MBHE (*node-splittingMBHE*)

O algoritmo 18 é utilizado pelo MBHE (algoritmo 5), para a efetuar a transformação da rede.

Algoritmo 18: Algoritmo de divisão dos nós utilizado pelo MBHE (doSplittingMBHE)

Dados: Grafo $G = (V,A)$, nó de origem s, nó de destino t, custo de cada arco l_k com $k \in A$, caminho semente p_s de s para t
Resultado: Grafo modificado $G' = (V',A')$

1 Criar o arco simétrico de cada arco do caminho semente, caso não exista.
2 Dividir cada nó do caminho semente (exceto o nó de origem e o de destino) em dois "sub-nós" (um de índice ' e outro de índice ") ligados entre si por dois arcos simétricos.
3 Ligar a cabeça de cada arco externo originalmente tendo como cabeça um nó do caminho semente, ao "sub-nó" de índice '. Ligar a cauda de cada arco externo originalmente tendo como cauda um nó do caminho semente, ao "sub-nó" de índice ".
4 Atribuir custo y aos arcos com direção do nó origem para o nó destino que ligam os "sub-nós" criados (ou seja, o arco que representa o nó no caminho semente p_s). Atribuir custo 0 aos arcos simétricos desses.
5 Atribuir custo $x + l_k$ aos arcos do caminho p_s, com custo original l_k. Aos seus arcos simétricos (que devem ser criados, se não existirem) atribuir o custo simétrico dos arcos correspondentes no caminho p_s.
6 Atribuir aos restantes arcos k que têm SRLG em comum com os arcos de p_s o custo $x + l_k$.

Bibliografia

[1] R. Bhandari. *Survivable Networks, Algorithms for Diverse Routing*. Kluwer Academic Publishers, Norwell, Massachusetts, USA, 1999.

[2] A. Farrel, J.-P. Vasseur, and J. Ash. A path computation element (PCE)-based architecture. IETF RFC 4655, 2006.

[3] T. Gomes and L. Fernandes. Obtaining a SRLG-disjoint path pair of min-sum cost. In Jacek Rak, David Tipper, and Krzysztof Walkowiak, editors, *RNDM 2010 – 2nd International Workshop on Reliable Networks Design and Modeling, colocated with ICUMT 2010*, number ISBN: 978-I-4244-7283-3, pages 116–122, Moscow, October 2010.

[4] T. Gomes, L. Jorge, P. Melo, R. Girão-Silva, and S. Mendes. Calculating a maximally node and SRLG-disjoint path pair of min-sum cost in GMPLS networks. In T. Cinkler, J. Tapolcai, and P.-H. Ho, editors, *9th International Conference on Design of Reliable Communication Networks (DRCN 2013)*, Budapest, Hungary, 4-7 March 2013. Accepted for publication.

[5] T. Gomes, C. Simões, and L. Fernandes. Resilient routing in optical networks using SRLG-disjoint path pairs of min-sum cost. *Telecommunication Systems Journal*, pages 1–13, 2011. on-line first.

[6] J. Q. Hu. Diverse routing in optical mesh networks. *IEEE Transactions on Communications*, 51(3):489–494, March 2003.

[7] IBM. Ilog cplex optimizer. http://www-01.ibm.com/software/integration/optimization/cplex-optimizer/.

[8] R.T. Marler and J.S. Arora. Survey of multi-objective optimization methods for engineering. *Structural and Multidisciplinary Optimization*, 26(6):369–395, 2004.

[9] E. Martins, M. Pascoal, and J. Santos. An algorithm for ranking loopless paths. Technical Report 99/007, CISUC, 1999.

[10] E. Martins, M. Pascoal, and J. Santos. Deviation algorithms for ranking shortest paths. *International Journal of Foundations of Computer Science*, 10(3):247–263, 1999.

[11] M. J. Rostami, S. Khorsandi, and A. A. Khodaparast. CoSE: A SRLG-disjoint routing algorithm. In *Proceedings of the Fourth European Conference on Universal Multiservice Networks (ECUMN'07)*, Toulouse, France, 2007.

[12] SNDlib. Survivable fixed telecommunication network design library. http://sndlib.zib.de.

[13] Miguel F. M. Soares. Cálculo de caminhos disjuntos nos nós e nos SRLG, de custo aditivo mínimo, para utilização num PCE. Master's thesis, University of Coimbra, 2012.

[14] J. W. Suurballe. Disjoint paths in networks. *Networks*, 4:125–145, 1974.

[15] A. Todimala and B. Ramamurthy. IMSH: An iterative heuristic for SRLG diverse routing in WDM mesh networks. In *13th International Conference on Computer Communications and Networks, ICCCN'2004*, pages 199–204, October 2004.

[16] J.-P. Vasseur, M. Pickavet, and P. Demeester. *Network Recovery – Protection and Restoration of optical, SONET-SDH, IP, and MPLS*. Elsevier, 2004.

[17] D. Xu, Y. Xiong, C. Qiao, and G. Li. Trap avoidance and protection schemes in networks with shared risk link groups. *Journal of Lightwave Technology*, 21(11):2683–2693, November 2003.

[18] J. Y. Yen. Finding the *k* shortest loopless paths in a network. *Management Science*, 17(11):712–716, July 1971.

[19] M. Żotkiewicz, W. Ben-Ameur, and M. Pioró. Finding failure-disjoint paths for path diversity protection in communication networks. *Communications Letters, IEEE*, 14(8):776 –778, August 2010.